EVENTSAPP

GESTIÓN AVANZADA DE CONTACTOS Y EVENTOS EN TERMINALES MÓVILES.

Titulación: Ingeniería Técnica en Informática de Gestión

Autor: Fernando de la Cuerda Gómez.

Tutor: Miguel Ángel Patricio.

Índice

Agradecimientos:

A mi familia, por estar siempre ahí y darme el apoyo y medios necesarios para seguir siempre adelante.

A mi rubia, por ser quien me ayuda y motiva para poder alcanzar y cerrar cualquier proyecto que se me presente.

Glosario:

EventsApp	Aplicativo software orientado a la gestión de la información de contactos y realizar operaciones de búsqueda, filtraje o tratamiento con estos.
Proyecto	Es la agrupación de diferentes fases y evolutivos que componen un producto determinado. Incluiríamos también en este concepto toda documentación o actividad que se relacionase con este producto.
Fase	Se trata de una segmentación lógica del proyecto. Cuando un proyecto tiene gran alcance se divide en fases para llevar un control del estado del mismo. En el caso de *EventsApp* el proyecto cuenta inicialmente con una Fase I aunque se ha planificado y planteado una Fase II futura.
Evolutivo	Un evolutivo suele ser un paso intermedio entre fases. Suelen surgir a raíz de la finalización de una fase o cuando una fase ya esté en periodo de desarrollo. Los evolutivos son funcionalidades o características añadidas o modificadas que se plantean a modo de nuevos requisitos ante necesidades surgidas en la implantación de una fase. Suelen ser pequeñas mejoras de gran alcance que requieren de una planificación y presupuesto propio.
Metadato	Los metadatos son atributos sobre objetos. Es decir, es información añadida que se le ofrece a un ente, lo cual permite su modelado particular. Si el sistema permite una gestión de metadatos dinámicos entonces nos encontraremos ante la posibilidad de modelar de forma unívoca cualquier elemento, a la par que podremos modelar de forma grupal un conjunto de elementos. Este concepto es muy utilizado en sistemas de alto nivel para el tratamiento de mucha información y es algo de lo que suelen carecer los sistemas actuales de gestión de contactos en terminales móviles.
Android	Sistema Operativo respaldado por Google, basado en Linux. Inicialmente creado para terminales móviles pero que se ha expandido a Tablet, Televisores, Mp3 u otros. Lo que lo diferencia con la competencia cómo IOs es su distribución libre y su código basado en Java.

Dalvik	Se trata de una variación del popular lenguaje de desarrollo Java. Incluido en la plataforma Android, su facilidad de uso y su desarrollo en entornos potentes cómo Eclipse o Android Studio permiten a la comunidad de desarrolladores hacer aplicativos con cierta facilidad.
Eclipse/ Android Studio	Se tratan de plataformas para el desarrollo de software. Eclipse con más experiencia pero también adaptada, ambas son perfectamente útiles para la creación de aplicativos.
IOs	Sistema Operativo de Apple. Basa su desarrollo en Objetive C.
PhoneGap	Se trata de una framework para el desarrollo de aplicaciones híbridas. Se basa en código HTML5, JavaScript y CSS3. Sus compilaciones sirven para ejecutarse en cualquier terminal móvil, conectándose incluso al API de estos.
Xamarín	Tecnología basada en C# para el desarrollo de aplicativos móviles en Android, Windows Phone e IOs.
VozIP	Se trata de un conjunto de características que permiten realizar comunicaciones de voz mediante un Protocolo de Internet.
Búsqueda Caliente	Se trata de un tipo de acción por la que un usuario quiere filtrar una lista de elementos según vaya introduciendo texto. Se parte de una lista completa que, según se vaya filtrando, la lista irá cambiando su tamaño y contenido de forma inmediata. Por tanto, este tipo de búsqueda permite eliminar la aparición de botones secundarios que confirmen la búsqueda. Es una metodología de búsqueda, orientada a la usabilidad que está muy presente en este tipo de aplicaciones.
Modelado de un contacto	Se trata de especificar, para un contacto dado, todo tipo de características que lo definan. Este conjunto de características deben aportar una imagen del contacto que lo diferencien del resto. Una o varias características pueden ser compartidas entre contactos, pero la combinatoria del conjunto de características de un contacto debe ser única para poder identificarlo unívocamente.
BDManager.java	Se trata de una clase java, implementada para este proyecto y que sirve de puerta de enlace entre una aplicación Android y una base de datos SQLite. Tiene los métodos básicos de este tipo de clases cómo GetCursor, ExecuteQuery o CreateBD, los cuales permiten las acciones primitivas con una BBDD. Los objetos de

	datos (aquellas clases que hacen referencia directa a una tabla) acceden a esta clase genérica para realizar las acciones oportunas e interpretar los datos que de esta se recojan. Por otro lado, la clase cuenta con patrones de diseño como el Singleton lo que la hace notablemente más usable.
Patrón Singleton	Se trata de un patrón de diseño por el que hay una sola instancia del objeto y esta se crea bajo demanda. Este patrón es muy útil ya que abstrae al programador de la inicialización del objeto y puede utilizarlo en cualquier momento teniendo la certeza de que estará creado e inicializado cuando este lo requiera.
DAO	Del inglés, *Data Access Object.* Se tratan de objetos que manejan la base de datos. Suelen ser tablas o vistas de la propia base de datos y ofrecen un conjunto de propiedades y métodos para interactuar con dicha base de datos.

1.- Introducción:

1.1.- Contexto:

La globalización, el acceso a la información y el avance de las telecomunicaciones han creado una motivación social por la que cada individuo se maneja y relaciona con un conjunto amplio de personas, estableciendo de este modo una red social por cada uno con relaciones con múltiples nodos y una gran cantidad de información de cada uno de ellos.

Este efecto es causa de nuevas problemáticas y necesidades enfocadas a la gestión de dichas redes y la información que de ellas se pueda derivar.

EventsApp surge para cubrir esta nueva necesidad, tratando de recopilar información de la red social de cada individuo y modelar la información que se considere oportuna para agilizar su tratamiento.

Esta herramienta software ayudar al usuario a gestionar los datos de sus contactos, incluyendo importante información añadida que permita un modelado personalizado.

Por otro lado, los terminales móviles tales como *smartphones o tablets* son sin duda uno de los grandes avances tecnológicos en lo que llevamos de siglo. El conjunto de funcionalidades que estos terminales ofrecen añadido a su condiciones físicas y comerciales han sido factores determinantes que han permitido que se integren en la vida de las personas de forma casi esencial.

EventsApp se ha adaptado a esta nueva ola tecnológica, basando su diseño, funcionalidad y potencial a los terminales móviles. No obstante, *EventsApp* va más allá de lo que es una aplicación móvil para convertirse en un concepto software de gestión de información.

1.2.- Objetivos:

A continuación se procede a enumerar el conjunto de objetivos que *EventsApp* trata de cubrir. Estos objetivos, orientados tanto a las necesidades físicas como conceptuales, dibujan a groso modo los intereses y motivaciones dadas para el desarrollo de este aplicativo.

Objetivo 1.- Desarrollo de un software fácil e intuitivo, orientado 100% a la usabilidad. Debe primar ante todo una interactuación con el aplicativo sencilla. El usuario no debe tener una formación especializada; el aplicativo debe seguir unos estándares de usabilidad que ayuden a guiar al usuario desde el momento cero.

Objetivo 2.- Gestión de contactos y redes de contactos. Debemos dar máxima importancia a cada contacto, orientando el objeto del aplicativo a la modelación de los datos de cada contacto y a las operaciones que con estos podamos realizar.

Objetivo 3.- Inclusión de nuevas funcionalidades que den valor añadido. Estas funcionalidades las enfocamos en 3 fases diferentes del proyecto.

- Fase I. Tratamiento de eventos y notas
- Evolutivo I. Modelación de la información mediante metadatos.
- Fase II. Comunicaciones con otros usuarios.

Objetivo 4.- Desarrollo de App. Queremos que el aplicativo sea inicialmente en terminales móviles, no descartando su uso futuro en terminales Web pero considerando que inicialmente tiene más potencia su orientación móvil.

En definitiva, *EventsApp* cuenta con cuatro objetivos claramente definidos que podremos resumir en:

'EventsApp será una aplicación móvil enfocada a la gestión de información de los contactos de un individuo. Su potencial radicará en su facilidad de uso y una serie de funcionalidades que la otorguen un atractivo singular'.

1.3.- Medios:

Como el proyecto *EventsApp* se compone por diferentes fases y evolutivos, podemos hacer un estudio de los medios necesarios en cada apartado para su ejecución.

De este modo encontramos para la Fase I, la cual trata sobre el aplicativo en sí de terminales móviles, la necesidad de un terminal móvil con Sistema Operativo Android 2.3 o superior y con un espacio de almacenamiento variable (en base a la información que se quiera guardar en la App) que parta de los 5MB para la construcción básica de la Base de Datos, Layouts y contenido en general del aplicativo.

Para la Fase II, la cual se basa en las comunicaciones, se plantea un esquema basado en servidores Microsoft Azure debido a sus características y potencial. El desarrollo de este sistema requerirá, al menos, de una base de datos relacional y un servicio web que atienda a las peticiones de los terminales móviles. Cómo ayuda e introducción a estos servicios se adjunta a la memoria del proyecto el documento *Windows Azure*, el cual contiene un estudio sobre la herramienta de Microsoft para el tratamiento de servicios, bases de datos y cuentas de almacenamiento en la red.

1.4.- Estructura de la Memoria:

La memoria del proyecto se compone en diferentes apartados que a continuación pasamos a describir.

1.- Introducción:

En este apartado se habla en líneas generales de lo que quiere desarrollar el proyecto. Trata de explicar la necesidad, los objetivos a cumplir y los medios necesarios para cumplirlo.

2.- Estado del Arte:

Aquí se realiza un estudio del mercado actual. Por un lado, la influencia de los terminales móviles en la sociedad y por otro la motivación que ha llevado a cabo la elección del S.O. Android para el desarrollo del proyecto.

En este apartado finalmente se hace un estudio de la competencia al aplicativo para evaluar los pros y contras frente a estos y determinar si nuestro producto tiene alguna ventaja competitiva frente a sistemas ya implementados y puestos en producción.

3.- Manual de uso:

Se trata de un leve manual de usuario. El cliente podrá leer este manual cargado de imágenes y anotaciones para entender y saber utilizar el software del proyecto a la perfección.

4.- Espacio de requisitos:

Es una colección de requisitos de diferente índoles (funcionales, de sistema, usabilidad, diseño, etc.) que se acuerdan entre la parte cliente y la parte desarrolladora. En este apartado se definen todas las necesidades que el aplicativo debe cumplir y como debe hacerlo. Este documento serviría para firmar el proyecto y todo lo que aquí quedase reflejado sería lo que entraría en el desarrollo obligado.

5.- Desarrollo:

Es un apartado para el grupo de trabajo. Se trata de explicar la base de datos, la arquitectura del proyecto, las metodologías de implementación usadas, la orientación del proyecto, etc. Por último, está el apartado para el presupuesto en el que se definen las horas y costes de dicha aplicación.

6.- Plan de pruebas:

Aquí veremos el conjunto de pruebas realizadas para dar el OK al aplicativo. Se trata de examinar a conciencia cada apartado y probar cada una de sus funcionalidades. Un comportamiento fuera de lo esperado sería motivo de error y debería ser corregido.

7.- Conclusiones:

Finalmente se habla sobre lo que ha aportado el proyecto. Se hace un estudio sobre los objetivos marcados y su correspondiente cumplimiento.

2.- Estado del Arte:

2.1.- Penetración de los terminales móviles:

Desde que los primeros terminales móviles naciesen allá en la Segunda Guerra Mundial en las manos de Motorola con fines militares hasta el día de hoy que se han convertido en una herramienta casi imprescindible para la vida cotidiana de las personas, han pasado apenas 60-70 años.

Durante este periodo no se ha cambiado simplemente el ámbito al que van dirigidos los terminales móviles, sino que las capacidades que estos han adquirido cambian completamente el concepto inicial que de ellos se tenía; ya no se trata de una herramienta para la comunicación uno a uno de forma remota, sino que ahora estos terminales son una herramienta para las comunicaciones genéricas con el mundo además de una herramienta de ocio y de gestión de todo tipo de necesidad.

Las aplicaciones móviles se han convertido hoy día en una herramienta cotidiana que alcanzan y satisfacen casi cualquier necesidad tecnológica que un individuo pueda demandar. Gracias a sus características físicas y el avance tecnológico que han sufrido, estos terminales móviles contienen todo lo necesario para ofrecer un amplio abanico de soluciones de cualquier índole.

El avance actual en esta campo sigue dos ramas; por un lado el avance tecnológico, ofreciendo cada vez mayor potencial a estos terminales y por otro lado el avance software, elaborando aplicativos más útiles o sencillos de manejar y que cubran nuevas necesidades.

EventsApp va por esta segunda rama, tratando de dar soluciones que cubran un conjunto de necesidades hasta el momento descubiertas.

2.2.- Desarrollo: Elección de tecnología.

A la hora de inicial el proyecto se ha llevado a cabo un estudio del entorno a desarrollar. Era requisito que el aplicativo funcionase para terminales móviles pero, teniendo esto en cuenta, ¿qué tipo de desarrollo y sobre que plataforma deberemos trabajar?

El conjunto de diferentes sistemas operativos que tienen los terminales móviles crea la necesidad a la hora de desarrollar un aplicativo móvil de tener que implementarlo en múltiples plataformas, multiplicándose de este modo los costes y tiempos finales.

Es por ello que surgen los llamados lenguajes híbridos. En reglas generales, podemos decir que estos lenguajes son implementados en un solo lenguaje y su compilación sirve para el conjunto de dispositivos existentes.

Con esta tecnología se resuelve la problemática de tener que implementar un mismo código en varios lenguajes, facilitando al desarrollador no tener que conocer los múltiples sistemas y ahorrando de este modo tiempos y costes.

¿Por qué elegir un desarrollo híbrido frente a uno nativo? Las ventajas de los lenguajes híbridos son obvias pero a la hora de realizar un desarrollo potente puede que estas ventajas se vean eclipsadas por carencias tecnológicas.

PhoneGap es la principal framework que trabaja esta tecnología híbrida. Basada en código HTML5, JavaScript y CSS3 es capaz de crear compilaciones que se conecten al propio API del terminal (accediendo a elementos del terminal cómo el acelerómetro o la cámara).

Además, potentes frameworks como *JQuery Mobile* son utilizadas en esta tecnología lo que le da una gran variedad de funcionalidades y una amplia extensión en el conjunto de desarrolladores.

Por otro lado nos encontramos con *Xamarín,* una tecnología basada en C# capaz de crear aplicativos en Android, IOs y Windows Phone. Esta tecnología permite

al igual que *PhoneGap* el desarrollo de un aplicativo para diferentes plataformas con un código desarrollado en un único lenguaje.

Sin embargo, a diferencia de *PhoneGap* cuyo código no se integra con el propio sistema del aplicativo sino que es interpretado por este, *Xamarín* sí que usa los propios controles del sistema para sus desarrollos.

Xamarín es una framework desarrollada en C# que podría cubrir las carencias que presenta *PhoneGap.* Ahora bien, la compilación que de ellas resulta es un lenguaje autogenerado que será más sucio que un lenguaje nativo generado particularmente.

Nos encontramos en este punto en la necesidad de elegir una tecnología para nuestro desarrollo. En resumen podemos hacer una tabla que compare los entornos:

	PhoneGap	Xamarín	Nativo
Interfaz de usuario	X	X	X
Requisitos básicos de sistema (control de eventos, base de datos, conexión con Web Service)	X	X	X
Desarrollo único multiplataforma	X	X (principales)	
Optimización de acceso y uso de componentes del terminal			X
Requisitos avanzados de sistema (Servicios Host)			X
Optimización de código			X

En base a esta tabla y a las necesidades de nuestro sistema podemos sacar conclusiones de la tecnología a elegir; recordemos que *EventsApp* cuenta con varias fases y que, aunque para la fase inicial cualquiera de las tres tecnologías sería válida, para fases futuras encontraríamos carencias que nos supondría tener que retroceder en el desarrollo y cambiar la tecnología, por lo que es necesario su estudio y evaluación de forma previa.

Dado que los sistemas nativos son los únicos que nos permiten algunas características tecnológicas como los Servicios Host o implementaciones óptimas (lo que nos sirve para que nuestra aplicación sea validada por el equipo de Google, Apple o Microsoft en sus respectivas herramientas de distribución de apps), nos decantamos por la elección de aplicativos nativos. Además, con este modo, llegamos a estudiar de forma más detallada el funcionamiento del sistema operativo, comprendiendo sus fases y peculiaridades sin que ninguna interfaz de terceros nos oculte ninguna funcionalidad o propiedad.

Una vez decantados por el uso de aplicativos nativos, a la hora de elegir por cual empezar y por qué, deberemos tener en cuenta diferentes aspectos tales como:

- Distribución: En el apartado '2.3.- Sistema Operativo. Android' veremos en mayor detalle la cuota de mercado de los diferentes Sistemas Operativos en el territorio nacional. Android cuenta en estos momentos con más de un 80%, por lo que es el candidato óptimo.
- Precio: Android es un lenguaje libre y su publicación y desarrollo es relativamente barato en comparación a IOs o Windows Phone.
- Funcionalidad: Android, IOs y Windows Phone son sistemas perfectamente válidos para nuestro sistema.

En definitiva, elegimos Android como tecnología de desarrollo dado su potencial y su distribución en el mercado. Se trata de un lenguaje nativo que nos ofrece todo tipo de posibilidad en nuestro desarrollo, atendiendo no solo a la fase inicial sino a sus futuras fases.

2.3.- Sistema Operativo. Android.

Android es un Sistema Operativo basado en Linux. Se trata de un sistema multiplataforma y gratuito lo que hace que sea potencialmente atractivo para su inclusión en diferentes empresas.

Por otro lado, el desarrollo de aplicaciones se hace en el lenguaje *Dalvik;* una variación de Java. Este hecho hace que haya una gran comunidad de desarrolladores dispuestos a trabajar la tecnología puesto que es un lenguaje conocido, experimentado y con grandes posibilidades.

Android se posiciona gracias a estas características y, por supuesto, a un funcionamiento estable y óptimo, en la referencia en el sector en apenas pocos años. Su principal competencia, IOs, lo sigue remotamente por una clientela específica y fiel creada a partir de una trayectoria intachable por parte de la compañía, pero que, poco a poco, se va decantando por esta tecnología.

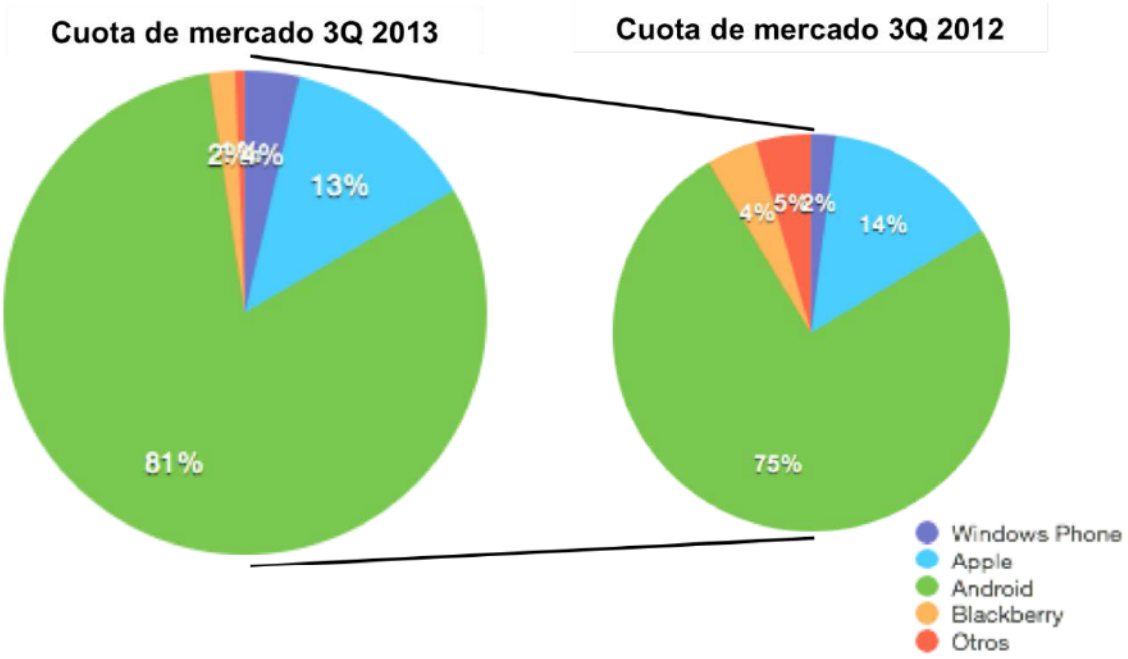


Ilustración 1. Cuota de mercado en la compra de terminales móviles 2012-2013

Hoy en día, ambas tecnologías (Android e IOs) se separan más por una competencia comercial y social que por las propias características que puedan ofrecen sus terminales. Android se posiciona como la alternativa de menor precio y mayor adaptación a los periféricos externos, contando además con un software libre para su desarrollo y basado en un lenguaje libre, conocido, experimentado y ya distribuido. IOs por su lado, se posiciona como la alternativa con más experiencia de usuario, ofreciendo interfaces más intuitivas y mayor robustez en sus sistemas. Buscan una separación argumentada en la calidad de producto, diseño e innovación, pero sus precios, atendiendo a sus componentes tecnológicos, son bastante superiores.

Por otro lado, otros sistemas impulsados por Microsoft, BlackBerry u otros no han conseguir alcanzar cuotas de mercado debido a peores decisiones comerciales o estratégicas en el ciclo de vida de sus productos. Es el caso por ejemplo de BlackBerry, cuya cuota de mercado se ha visto fuertemente afecta pasando de ser una de los principales suministradores de terminales móviles al sector empresarial a estar casi extinguido.

Por otro lado, Microsoft cometió varios errores estratégicos cómo la creación de múltiples plataformas (Windows Mobile y Windows Phone), ambas de características muy diferentes. Quiso meterse en el mercado gracias a costosas campañas de promoción y publicitarias pero se encontró con una clientela ya posicionada y sus esfuerzos no consiguieron dar grandes resultados.

En definitiva, Android se posiciona en la actualidad como el principal Sistema Operativo para terminales móviles si atendemos a las cuotas de mercado. Este posicionamiento es gracias a las características técnicas del sistema que le otorgan una gran potencia y bajo coste. Su competencia, principalmente IOs, gestiona una clientela específica que se argumenta en los pilares principales de la compañía Apple que son calidad, diseño y robustez.

2.4.- Mercado Competente:

En la actualidad hay un gran número de aplicaciones móviles que gestionan la información de los contactos del usuario de forma ordenada y eficiente. Además, cuentan con grandes equipos e infraestructuras que permiten a estas aplicaciones estar actualizadas y ofrecer servicios de integración, backup y diseño difícilmente alcanzables por aplicaciones realizadas por particulares.

No obstante, *EventsApp* no trata de cubrir esta necesidad genérica de gestión de contactos sino que quiere orientar su éxito al conjunto de funcionalidades añadidas que, por el momento, ninguna competencia ha desarrollado y difundido en la comunidad de aplicativos móviles.

Ilustración 2. Aplicaciones a estudiar.

Realizando un estudio de mercado podemos encontrar multitud de aplicativos cuya funcionalidad sea la gestión de contactos. Haciendo filtros por características técnicas, potencial, diseño o expansión, nos quedamos con la siguiente lista de aplicativos los cuales pasaremos a estudiar en detalle a continuación:

2.4.1.- Plaxo:

Se trata de un aplicativo Web capaz de sincronizarse con los terminales móviles de forma bidireccional (bajo pago). Permite la sincronización con diferentes cuentas y el tratamiento de datos por grupos, así como compartir información de fichas de contactos. Además ofrece diferentes paquetes con grandes utilidades cómo asistencia personal y 'Actualizaciones inteligentes' que hacen que la información de los contactos se mantenga actualizada estudiando el estado de los mismos en las diferentes cuentas que estén sincronizados.

Podemos encontrar más información en http://www.plaxo.com/about/plaxo.

2.4.2.- Evernote Hello:

Es una aplicación móvil que basa su funcionamiento en la creación de fichas personales que contienen información amplia de los contactos. Basado en una interfaz muy atractiva y una comunicación en la nube, tiene un funcionamiento óptimo. Tiene una orientación, a grandes rasgos, que se precipita más a tener una foto temporal de un contacto, abarcando información sobre este del tipo imagen, anotaciones, información de comunicación con este (teléfonos, emails), que a lo que sería propiamente modelar al contacto. No contiene un sistema de chat ya que su orientación se queda en la gestión de información.

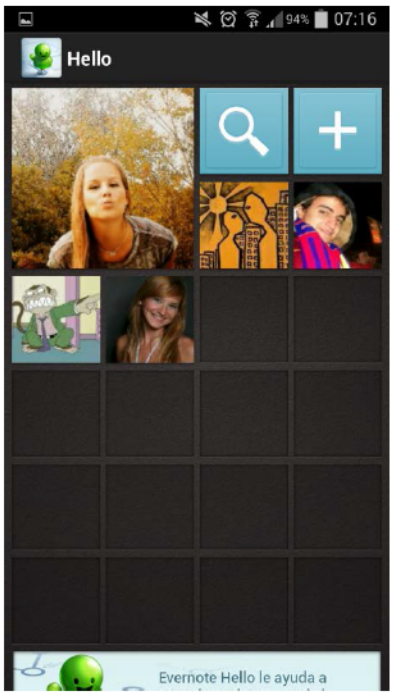


Ilustración 3. Pantalla inicio Evernote Hello

Podemos encontrar más información en http://evernote.com/intl/es/hello/

2.4.3.- ExDialer & Contacts:

Es un aplicativo orientado a las llamadas. Tiene una interfaz muy sencilla por la que gestiona al conjunto de contactos y el registro de llamadas. Puede añadir anotaciones y un conjunto de atributos predefinidos.

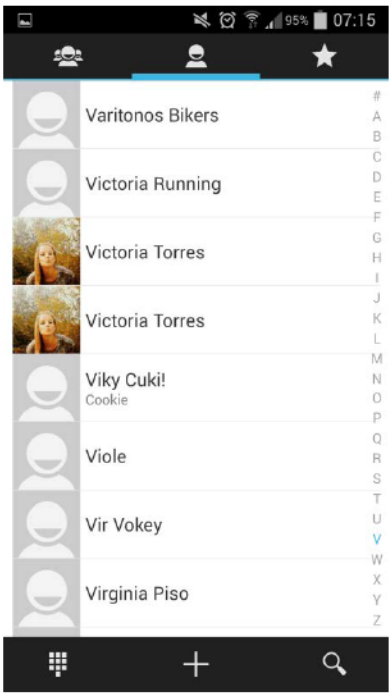


Ilustración 4.- Pantalla de inicio de ExDialer & Contacts

Una de las características mas destacadas es la posibilidad de realizar llamadas mediante VozIP, lo que la hace atractiva frente a sus competidoras.

2.4.3.- Android. Contactos:

Esta es la propia agenda de teléfonos del terminal. Ofrece la capacidad de visualizar los contactos, hacer un conjunto de favoritos y agrupaciones. Además, permite la inclusión de datos básicos preestablecidos como eventos, notas, melodías de llamada, etc.

Su integración con el terminal móvil es óptima y tiene una plataforma en la nube que permite independizar del terminal la agenda para salvaguardar la información que esta contuviese.

Está ligado a una cuenta de Google que previamente deberemos tener pero para el uso de estos terminales es un requisito casi imprescindible.

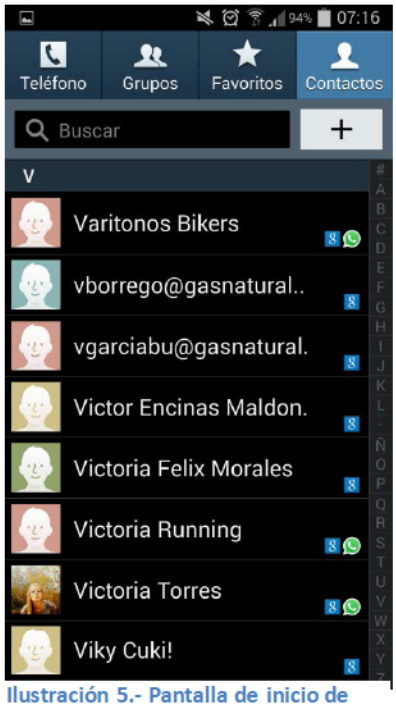


Ilustración 5.- Pantalla de inicio de Android. Contactos

2.4.4.- EventsApp:

Esta aplicación, en su primera fase de desarrollo, cuenta con una gestión sencilla de los contactos que permite configurarlos en pocos pasos. Además, en fases futuras se prevé la modelación de la información de forma no vista hasta el momento lo que permite una gestión única de la información con un mayor número de funcionalidades como el filtrado complejo.

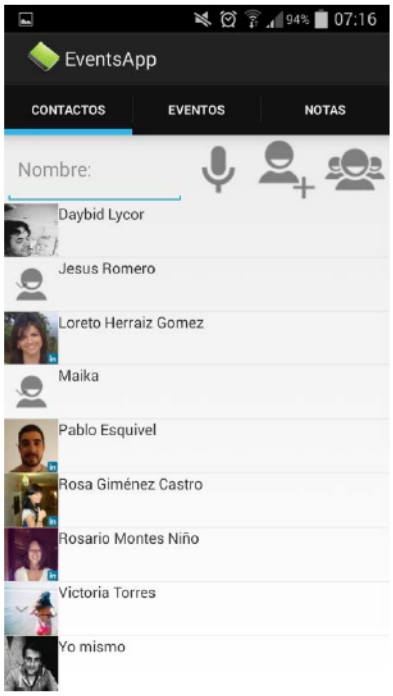


Ilustración 6. Pantalla de inicio de EventsApp

2.4.5.- Comparativa funcional:

A continuación vamos a pasar a detallar los pasos que tendríamos que realizar para hacer las mismas funcionalidades en los diferentes aplicativos. Veremos lo complejo o no que resulta en cada software y evaluaremos su nivel de calidad. Más tarde, en el punto *'2.4.5.- Crítica personal'* haremos una crítica personal sobre el conjunto de características para ver con que software nos quedaríamos y las razones.

2.4.5.1.- Acceso al aplicativo:

Este caso es el mas sencillo de todos. Simplemente queremos acceder al aplicativo para visualizar las funcionalidades que nos ofrece.

CardsHaring by Plaxo:

Nos aparece una pantalla para logarnos en 'Google', 'Facebook' o crear nuestra propia cuenta. Intentamos realizar cualquiera de estos accesos pero nos da diferentes errores. Aunque podemos acceder a su plataforma Web, al no poder acceder al aplicativo móvil debemos dejarlo de lado en el estudio de competencia ya que no cumple uno de los requisitos básicos del documento que es su gestión mediante terminales móviles.

Evernote Hello:

Tras habernos logado en su sistema, accedemos correctamente a la pantalla principal del aplicativo. Nos encontramos con el conjunto de usuarios que hemos introducido ordenados por orden de inserción inverso, es decir, aparecen los últimos incorporados en el aplicativo. Nos aparece la imagen del contacto pero no nos aparece ninguna información de este en una primera instancia (a no ser que el contacto no tenga imagen en cuyo caso si que se nos describe al contacto). Tiene una capacidad de 19 contactos en pantalla simultáneamente.

ExDialer & Contacts:

Al instalarnos la aplicación se nos crean en el terminal dos accesos directos. Por un lado, estaría la propia agenda de teléfonos y, por otro lado, la gestión de llamadas. Ambos accesos directos dirigen al mismo aplicativo móvil, solo que a diferentes apartados del mismo.

Este modelo es el mismo que utiliza *Android. Contactos,* permitiendo al usuario acceder desde fuera a las principales funcionalidades del aplicativo de forma separada.

Independientemente del acceso por el que entres, el aplicativo según te conectas contiene el conjunto de contactos de la agenda y el registro de llamadas del terminal ya importado. No ha sido necesario darse de alta en ningún sistema ni hacer ninguna gestión previa, lo cual es muy atractivo para el usuario.

Encontramos, por cada contacto, una imagen y un texto que representa su nombre y apellidos. En la pantalla podemos visualizar hasta 9 contactos simultáneamente.

Android. Contactos:

Al pulsar sobre el icono nos presenta en nuestra agenda de teléfono. Encontramos por cada contacto un pequeño icono y un texto que nos describe su nombre y apellidos. Tiene una capacidad de hasta 9 contactos simultáneamente.

EventsApp:

Cuando abrimos el aplicativo se nos presenta una pantalla con hasta 9 contactos. Cada contacto se identifica por un pequeño icono, nombre y apellidos.

En un principio la lista está vacía, por lo que tendremos que añadir contactos manualmente o importarlos de alguna cuenta que tenga disponible.

2.4.5.2.- Buscar un contacto:

Esta acción es una de las más importantes a tener en cuenta. El objeto principal del proyecto es la gestión de contactos, por lo que la búsqueda y tratamiento del contacto deben ser los pilares fuertes del aplicativo y su competencia.

Evernote Hello:

Para acceder a las búsquedas hay que cambiar de pantalla, lo cual puede perder al usuario. Requiere además que el usuario inserte algo de texto para que la búsqueda se inicie.

En el lado positivo destacar que la búsqueda se hace sobre los contactos del terminal, no solo de los que hayas importado. Es un aspecto que *EventApp* por ejemplo no ha querido tener en cuenta pero que no se descarta para algún evolutivo futuro. Además, la búsqueda que hace *Evernote Hello* indica el origen de la información (agenda de teléfonos o propia app) lo cual es muy útil para la gestión de contactos.

ExDialer & Contacts:

La búsqueda se hace en caliente sobre la propia lista de contactos. Gestiona correctamente de este modo la usabilidad del aplicativo permitiendo al usuario no perder de vista el listado de contactos y manteniendo los roles preestablecidos en este tipo de aplicaciones.

Android. Contactos:

Al igual que *ExDialer & Contacts* o *EventsApp*, esta aplicación tiene una caja de texto para hacer búsqueda caliente sobre la lista de contactos. El usuario no pierde en ningún momento de vista sus contactos y se van filtrando dinámicamente según se vaya introduciendo texto.

EventsApp:

EventsApp sigue la misma política de usabilidad que *ExDialer & Contacts* y *Android. Contactos.* Esta política la cumple gracias a una caja de texto en la parte superior de la lista de contactos donde el usuario según va metiendo texto se va haciendo un filtro en tiempo real sobre los contactos.

Por su parte, *EventsApp* incorpora un icono de micrófono para la búsqueda por voz. Es un icono grande que no pasa desapercibido y puede ser útil para cuando el usuario no pueda manejar el teclado fácilmente.

2.4.5.3- Modelado de un contacto:

En este apartado tenemos en cuenta todo aquello relacionado con la gestión de información de cada contacto. No solo tenemos en cuenta el alcance o adaptabilidad de la herramienta sino también su diseño o facilidad de uso.

Evernote Hello:

Nos permite información básica: nombre y apellidos, imagen de contacto, lista de teléfonos y lista de emails. Además nos permite añadir un título, una organización o un Twitter.

Por otro lado, nos permite gestionar notas del contacto, fotos y eventos. Aunque el manejo de esta información nos pierde debido al alto número de elementos que aparecen en pantalla.

ExDialer & Contacts:

No tiene una gestión personalizada del contacto. Su edición te remite a la propia agenda del teléfono por lo que sus posibilidades dependen de la evolución de terceros.

Android. Contactos:

Permite el tratamiento básico de información (nombre y apellidos, imagen, emails y teléfonos). Además, incorpora un conjunto de atributos predeterminados a señalar como: Fonética, Notas, Dirección, Eventos, Relación, Alias…

EventsApp:

Permite el tratamiento básico de información al igual que el resto (nombre y apellidos, imagen, emails y teléfonos). Además también permite la incorporación de notas y eventos en cada contacto.

El primer evolutivo sobre el proyecto además añadirá la gestión avanzada de metadatos, lo que permitirá añadir a cada contacto información personalizada que lo diferencia del resto y permita hacer búsquedas avanzadas sobre los contactos.

2.4.6.- Crítica personal:

Lo primero que me llama la atención es que tanto *Plaxo* como *Evernote Hello* presentan su interfaz gráfica en inglés, lo cual limita enormemente el número de usuarios que puedan utilizar estas herramientas. *ExDialer & Contacts, EventsApp* y *Android. Contactos* definen la lengua del aplicativo en base al lenguaje del terminal.

Por otro lado, tanto *Plaxo* como *Evernote Hello* requieren de un registro previo para el uso del aplicativo. ¿y si no tenemos conexión a internet? ¿y si no queremos

darnos de alta en ningún sistema externo?. La necesidad de registro limita también el número de usuarios. Pensemos que el cliente podría querer tener el aplicativo en local, guardar algún contacto de forma casual, gestionar pequeñas cantidades de forma confidencial, etc.

La sencillez es algo que debe estar presente en estos tipos de aplicativos ya que el mercado destino a ser cliente del aplicativo recoge cualquier modelo de persona; desde personas con estudios y formación tecnológica hasta gente que no alcance conocimientos avanzados sobre aplicativos móviles. En este apartado, *ExDialer & Contacts* se queda incluso demasiado corta debido a su interfaz demasiado básica. El usuario sólo elegirá está aplicación si tiene claro que desea realizar llamadas por VozIP, dado que de no ser así, la interfaz tan básica le creará una sensación de baja calidad.

Centrándonos en este apartado, el aplicativo que probablemente tenga la mejor apariencia visual es *Evernote Hello,* aunque no sigue los cánones generales de agendas telefónicas. Esto quiere decir que mucha funcionalidad ya implantada en los usuarios por aplicaciones anteriores se puede perder.

El hecho de que por lo general, las agendas de teléfono electrónicas se basen visualmente en una lista de contactos, ordenada de forma vertical, con una imagen en la parte izquierda de cada contacto e información de este en la parte derecha, es un protocolo, no escrito, que define no ya solo un concepto de 'Agenda', sino una forma de trabajo y uso que permite a las aplicaciones tener la certeza que el usuario va a saber hacer uso de ellas.

Parece un aspecto poco relevante pero el hecho de que un usuario sepa de antemano manejarse en un aplicativo a desarrollar es una baza muy importante a tener en cuenta. Recordemos que este tipo de aplicaciones no deberían llevar ningún tipo de formación adicional, por lo que cuanto menos tenga que molestarse el usuario en aprender el uso del aplicativo, más probabilidades de éxito tendremos.

En líneas generales todas estas aplicaciones las 4 aplicaciones estudiadas cumplen con los requisitos básicos del proyecto que son la gestión de información de contactos. Ahora bien, a la hora de decantarnos por alguna, tenemos en cuenta no ya solo los requisitos mínimos, sino también aspectos como la usabilidad, diseño o alcance.

Por esta vía *EventsApp* se hace notar gracias a una interfaz muy básica y elegante que sigue los patrones de usabilidad en este tipo de aplicaciones. Además, su funcionalidad amplia la de la competencia no solo permitiendo la comunicación con las acciones de

forma más sencilla o intuitiva, sino también ofreciendo nuevos conceptos que pueden ser muy interesantes como la gestión de metadatos (Evolutivo I), comunicación entre usuarios (Fase II), uso compartido de eventos y notas entre contactos de la agenda (Fase I) o selección personal de la información a tratar (Fase I).

En definitiva, *EventsApp* cumple los requisitos mínimos de gestión de contactos al igual que sus competentes. Ahora bien, ofrece una serie de ayudas y abanico de posibilidades que lo distinguen.

2.4.7. EventsApp, distinción del aplicativo:

Como llevamos tiempo comentando, *EventsApp* contiene un conjunto de herramientas o funcionalidades combinadas con un diseño atractivo que permite su distinción en el mercado. A continuación pasaremos al estudio detallado de los pilares que hacen que *EventsApp* pueda triunfar en un mercado ya atendido.

Diseño:

EventsApp tiene un diseño muy sencillo que cumple con los patrones de usabilidad de este tipo de aplicaciones. Además, que toda funcionalidad se gestione mediante el uso de iconos y que se haya eliminado del aplicativo opciones tipo 'Menú' permiten navegar fácilmente por la aplicación.

EventsApp tiene tres conceptos claramente diferenciados: Contactos, Eventos y Notas. Estos conceptos permanecen siempre visibles y accesibles en los menús principales de la aplicación gracias a una barra de pestañas.

En definitiva, *EventsApp* tiene siempre visible las características principales del aplicativo. Además, estas características se pueden gestionar fácilmente mediante el uso de iconos lo que permite, de forma visual, navegar fácilmente por el aplicativo.

Selección personalizada de contactos:

Por lo general, las aplicaciones de este tipo dan la posibilidad de conectarse a una fuente externa de información, importando los contactos que de ahí se puedan sacar. *EventsApp* va un paso más allá, permitiendo al usuario seleccionar de forma personalizada la información a importar.

En la actualidad, una persona tiene un grupo muy extenso de contactos pero, probablemente, solo gestione usualmente un subgrupo de estos.

EventsApp no importa automáticamente todos los contactos; permite al usuario seleccionar los contactos a importar.

Relación contacto-evento-nota:

Las aplicaciones en la actualidad permiten incluir información de tipo nota o evento sobre contactos. Ahora bien, la posibilidad de relacionar notas a un evento ya no la encontramos en estas aplicaciones. Por ejemplo, si un evento es acudir a una reunión determinada. Podemos poner notas del tipo 'recordatorio de material a llevar', 'resumen de la reunión', etc. Además, una nota nos puede servir para varios eventos, por lo que podremos unirla fácilmente a un conjunto de eventos.

Por otro lado, un contacto puede tener un número indefinidos de notas. Estas notas, además, pueden ser válidas entre varios contactos (este concepto se sustituirá por metadatos en la Fase I del proyecto).

Uso de metadatos (Evolutivo I):

EventsApp, en su primer evolutivo, permitirá una gestión avanzada sobre la información de los contactos. El modelado de cada contacto se podrá hacer de forma personalizada añadiendo los conceptos de *Tipo de Metadato* y *Metadato.*

Con estos conceptos podemos modelar de forma única a cada contacto e incluir cualquier tipo de información que creamos oportuna. Esta información es importante señalar que es acorde a las necesidades del usuario y no a un subconjunto preestablecido de opciones que puede que no recojan las necesidades del usuario.

La diferencia principal entre modelar un contacto o añadirle notas es que el modelado está tipado y permite posteriormente una búsqueda avanzada sobre conceptos, cosa que por notas no nos sería posible.

Comunicación entre usuarios (Fase II):

Para la Fase II se ha planificado esta funcionalidad. Aunque ya hay aplicativos parecidos que comparten información o mensajes como *WhatsApp o Line, EventsApp* no se dirige a las conversaciones tipo 'Chat', sino que se orienta a una comunicación de información y a un modelado común de la información. Es decir, se trata de que un conjunto de usuarios compartan y modelen información acerca de contactos, eventos o notas.

La competencia, en la actualidad, no permite llegar a estos límites de modelado. Generalmente están capados por un conjunto de opciones preestablecidas

que impiden que el usuario pueda tener una participación activa en la gestión de información compartida por nuestros contactos.

2.4.8.- Tabla de comparación:

A continuación pasamos a detallar las conclusiones en base al estudio sobre estas aplicaciones. No solo nos vamos a centrar en sus características técnicas sino también vamos a realizar una crítica fundamentada en la usabilidad, diseño o experiencia de usuario que encontremos.

	Evernote Hello	ExDialer & Contacts	Contactos	EventsApp
Listado de contactos	X	X	X	X
Tiene aplicativo móvil	X	X	X	X
Uso sin autentificación		X	X	X
Gestión básica de contactos	X	X	X	X
Gestión de notas sobre contactos	X	X	X	X
Gestión de nota sobre múltiples contactos				X
Gestión de eventos sobre contactos	X		X	X
Inclusión de múltiples contactos en eventos	X		X	X
Notas en eventos				X
Búsqueda caliente de contactos		X	X	X
Búsqueda caliente por voz				X
Llamada rápida (en 3 o menos clicks)		X	X	X
Listado de eventos				X
Búsqueda de eventos				X
Listado de notas				X
Búsqueda de notas				X
Selección personal de información a importar				X
Unión de contactos			X	X
Mínimo nº de pantallas en todo el aplicativo (entre 1-5 pantallas diferentes)				X
Uso total de componentes del aplicativo (sin utilizar usos de terceros)	X		X	X
Patrones de usabilidad		X	X	X

3.- Espacio de Requisitos:

En este apartado se definirán en detalle los requisitos del aplicativo. Estos requisitos los podemos separar en cuatro grandes bloques:

- Requisitos funcionales: Referido a todas aquellas acciones que el sistema debe ser capaz de realizar.
- Requisitos de sistema: En este apartado agruparemos las necesidades técnicas y físicas.
- Requisitos de usabilidad: Aunque es una derivada de los funcionales, la separamos por su importancia. Aquí encontraremos las necesidades de diseño o tratamiento. Es decir, no contestan al 'QUÉ', sino al 'CÓMO'.
- Requisitos de desarrollo: Se trata de definir no solo el producto sino como realizar la implementación del mismo; aquellas normas, patrones o conjunto de reglas que deberemos seguir para su desarrollo.

En base a este documento se realizar el presupuesto. Toda anotación, funcionalidad, concepto o herramienta que no quede aquí descrita saldrá del ámbito del proyecto y no podrá ser reclamable.

Se requeriría la firma y aprobación de este documento por parte del cliente para continuar con el desarrollo del proyecto.

Se han definido no solo los requisitos de la Fase I sino también los requisitos del Evolutivo I y de la Fase II. Estos dos últimos proyectos tienen el plan de requisitos sin confirmar, pero se incluyen para dar una visión global del proyecto.

Cómo podremos observar, el documento de requisitos de la Fase I y Evolutivo I están mucho más detalladas que el de la Fase II. Esto se debe a que la Fase I y Evolutivo I están implantados y estudiados mientas que la Fase II simplemente está pensada. Aún no se ha hecho el plan de proyecto de dicha Fase II.

3.1.- Requisitos Fase I:

El conjunto de requisitos que aquí se nombran tratan de definir el proyecto inicial *EventsApp.* Los pilares de esta fase son la creación de un proyecto móvil para Android que pueda gestionar contactos, eventos y notas.

3.1.1.- Requisitos Funcionales:

RF1.- La aplicación tendrá 3 pantallas divididas en pestañas. Esas pantallas, de izquierda a derecha, contendrán los datos de 'Contactos', 'Eventos' y 'Notas'. Por defecto, la aplicación se inicializará en la pantalla de 'Contactos'.

RF2.- La pantalla de contactos tendrá una caja de texto donde se hará búsqueda caliente. Esta búsqueda actuará sobre el nombre y/o apellidos. No distinguirá entre mayúsculas y minúsculas.

RF3.- En la pantalla de contactos se podrá hacer búsqueda caliente por voz mediante la habilitación de un botón destinado a ello.

RF4.- Se podrán añadir tantos contactos como se desee. Será obligatorio introducir, al menos, nombre y apellidos.

RF5.- Los contactos se listarán ordenados alfabéticamente. Se mostrará, por cada contacto, una imagen, nombre y apellidos.

RF5.1.- Al pulsar brevemente sobre un contacto deberemos poder ver el detalle de dicho contacto, en donde deberá aparecer, al menos:

- Imagen, nombre y apellidos del contacto.
- Nº de teléfonos que tiene asociado
- Nº de Emails que tiene asociado
- Nº de Eventos en los que participa
- Nº de Notas en las que participa.

RF5.2.- Al pulsar prolongadamente sobre un contacto deberemos entrar en modo edición. Desde ahí, deberemos poder modificar, al menos, sus datos básicos (imagen, nombre y apellidos).

RF6.- Desde la pantalla de contactos, podremos asignar a cada contacto nº de teléfono, emails, eventos y notas. También debemos poder crear un nuevo registro de estos tipos de forma rápida y que se asigne directamente al contacto.

RF7.- Desde la pantalla de contactos podremos realizar llamadas y/o enviar emails. Necesitamos que el contacto tenga lógicamente un nº de teléfono y/o una cuenta de correo.

RF8.- Desde la pantalla de contactos podremos importar contactos desde otros sistemas. Se pide, al menos, que se importen desde la propia agenda de teléfonos y desde alguna red social.

RF8.1.- Los datos de importación mínimos deben ser Nombre y Apellidos. En caso de estar disponibles, se podrían descargar datos de Imagen, Nº de teléfono, Nº de email, Eventos y Notas.

RF8.2.- Durante la descarga de información, si esta tarda más de 5segundos, se deberá establecer un diálogo de espera con el usuario donde se le informe del estado de la descarga de contactos.

RF8.3.- No se deberán descargar todos los contactos de donde se conecte el usuario (agenda del teléfono y/o red social). El usuario, manualmente, seleccionará los contactos que quiere importar tras que el aplicativo le liste el conjunto de contactos que haya encontrado en dicha importación. Además, si un contacto ya lo tiene importado, se deberá reflejar de algún modo para que el usuario sepa que no debe importarlo de nuevo.

RF8.4.- Si un usuario en un momento m1 importa a un contacto y tras ello en un momento m2 vuelve a la pantalla de importación, verá que el contacto ya está importado. Si el usuario cancela esa importación para el contacto, el contacto NO se borrará del sistema; los contactos solo se podrán borrar de la pantalla de contactos pues es ahí donde el usuario puede corroborar los eventos, notas, teléfonos y emails que tiene asignado. En definitiva, los contactos solo se pueden eliminar desde la pantalla de contactos.

RF9.- Desde la pantalla de contactos se deberá poder realizar uniones de contactos de forma sencilla.

RF10.- La pantalla de eventos deberá tener dos cajas de textos para introducir fechas.

RF10.1.- Por defecto, las cajas de texto para inserción de fechas estarán vacías. Al pulsar sobre estas solo se podrán introducir fechas. El usuario no debe tener la posibilidad de introducir otra cosa.

RF10.2.- Si se introduce ninguna fecha, se listarán todos los eventos que haya registrados ordenados alfabéticamente.

RF10.3.- Si se introduce una fecha de inicio, se listarán aquellos eventos que estén en un periodo de ejecución superior o igual a dicha fecha.

RF10.4.- Si se introduce una fecha de fin, se listarán aquellos eventos que estén en un periodo de ejecución inferior o igual a dicha fecha.

RF10.5.- Si se introducen ambas fechas, se listarán aquellos eventos que estén en un periodo de ejecución comprendidos entre dichas fechas.

RF11.- En la pantalla de eventos, estarán todos los eventos ordenados cronológicamente y se mostrará, al menos, la fecha de inicio, la fecha de fin y el asunto del evento.

RF13.- Se podrán crear tantos eventos cómo se desee. Cada evento deberá tener, al menos, los siguientes campos:

- Asunto
- Descripción
- Fecha de inicio
- Fecha de fin
- Lugar (opcional)

RF14.- Al pinchar de forma sencilla en un evento deberemos poder ver su información de fechas de inicio, fin, asunto, descripción y lugar. También deberemos poder ver los contactos y notas asociados al evento.

RF15.- Al pinchar prolongadamente deberemos poder editar sus campos de asunto, descripción, lugar y fechas.

RF16.- Deberemos poder asociaciones de cada evento de forma sencilla con notas y contactos.

RF17.- Un evento solo se podrá eliminar desde esta pantalla.

RF18.- En la pantalla de notas tendremos una caja de texto donde el usuario podrá hacer búsqueda caliente sobre las notas. Esta búsqueda se hará respecto al título de cada nota.

RF18.1.- Inicialmente, la caja de búsqueda caliente estará vacía.

RF19.- En la pantalla de notas habrá una caja de texto donde solo se podrán introducir fechas.

RF19.1.- En la caja de texto destinada para fechas, al insertarse una fecha solo se listarán las notas para fechas iguales o superiores a la fecha introducida.

RF19.2.- Por defecto, la caja de texto de fechas estará vacía.

RF20.- Se podrá crear tantas notas cómo se desee. Las notas deberán tener los campos de:

- Asunto
- Descripcíon
- Fecha.

RF21.- Sólo se podrán eliminar notas desde la pantalla de notas.

RF22.- Se podrá relacionar a cada nota una serie de contactos y/o eventos de forma similar a como se hará en las pantallas de contactos y/o eventos.

RF23.- El aplicativo estará capacitado para mostrarse en varios idiomas. En caso de que un idioma no se haya definido, se pondrá por defecto el inglés.

3.1.2.- Requisitos de Sistema:

RS1.- El aplicativo se montará sobre un sistema operativo Android 3.0 o superior.

RS2.- El aplicativo requerirá almacenamiento interno en el terminal para la creación de su base de datos. Al menos necesitará 5MB de memoria del aplicativo.

RS3.- El aplicativo no necesitará de grandes exigencias de procesamiento, gráficos o potencia del terminal. En principio, cualquier terminal del mercado estará capacitado para la ejecución del aplicativo.

RS4.- El aplicativo no se diseña para ningún tipo de terminal en particular. Este aplicativo se podrá instalar y ejecutar en cualquier Smartphone o Tablet.

3.1.3.- Requisitos de Usabilidad:

RU1.- El diseño del aplicativo debe ser sobrio y elegante. No debe de tener dibujos o elementos que distraigan el objetivo del aplicativo. Recordemos que es un aplicativo de gestión o trabajo y no de entretenimiento.

RU2.- Los iconos utilizados en el aplicativo serán los oficiales de las librerías de Google, para adaptarse a las métricas de desarrollo que dictan y unificar las funcionalidades genéricas ya establecidas (por ejemplo, añadir contacto o añadir eventos).

RU3.- El botón menú no tendrá funcionalidad en el aplicativo. Esta aplicación se pretende difundir a otros terminales con distintos sistemas operativos (como IOs o Windows Phone) y este botón es específico del Sistema Android. Por tanto, todo método de usabilidad deberá quedar reflejado en la propia pantalla. De este modo, las diferencias de usabilidad con otros terminales serán las mínimas posibles.

RU4.- La metodología de trabajo en el aplicativo debe ser uniforme. Esto implica que para hacer acciones similares, la forma de realizarlas también debe ser similar (por ejemplo, añadir un contacto, evento o nota, visualizar un dato, etc).

3.1.4.- Requisitos de Desarrollo:

RD1.- Se debe realizar un código organizado por capas que permita la escalabilidad. Este proyecto es un proyecto abierto que se prevé ampliaciones próximas y que debe estar preparado para ello.

RD2.- Los accesos entre capas deben ser independientes y debe haber una forma de comunicación segura entre estas.

RD3.- En el acceso a datos, deberemos tener un conjunto de DAO que nos permita trabajar con la BBDD de forma segura e independiente. Los modelos podrán ser los propios DAO o clases independientes que accedan a estos DAO para el tratamiento de datos.

RD4.- El multilenguaje se debe desarrollar mediante recursos, siendo algo independiente al código en sí. En caso de que un recurso no exista para una lengua en particular deberá coger el de por defecto.

3.2.- Requisitos Evolutivo I:

El conjunto de requisitos que a continuación se detallan tratan de establecer pequeñas mejorar sobre la Fase I y la capacidad de modelar la información mediante el uso de metadatos y tipos de metadatos.

3.2.1.- Requisitos Funcionales:

RF1.- Se debe poder incluir imágenes en los conceptos de 'evento' y 'notas' tal y como se hace en la actualidad con los contactos.

RF2.- Habrá una pantalla para la gestión de las imágenes; actualmente se recoge un cuadrado correspondiente a la esquina superior izquierda de la imagen. Este cuadrado se debe poder modificar por el usuario.

RF2.1.- Al seleccionar la cámara para la captura de la imagen, se debe abrir una nueva actividad en donde se vea lo que está visualizando la cámara en real con un cuadrado en formato claro y el resto de la imagen en oscuro. Este cuadrado el usuario lo podrá mover a su antojo. Cuando guarde la imagen, se guardará simplemente la imagen del cuadrado en claro.

RF2.2.- Al igual que con la cámara deberá pasar al seleccionar una imagen de la galería. Un cuadrado en claro sobre un fondo oscurecido.

RF3.- Se incluirá una nueva funcionalidad de gestión de metadatos en la pantalla de contactos. Para ello se agruparán los iconos de 'Añadir contacto' e 'Importar contactos' en un nuevo icono 'Menú'. Al pulsar sobre este icono aparecerá una lista con las siguientes posibilidades:

- Añadir contacto (funcionalidad igual que la existente)
- Importar contactos de LinkedIn (funcionalidad igual que la existente)
- Importar contactos de la agenda (funcionalidad igual que la existente)
- Juntar contactos existentes (funcionalidad igual que la existente)
- Búsqueda avanzada (nueva funcionalidad)
- Gestión de metadatos de contactos (nueva funcionalidad)

RF3.1.- La gestión de metadatos en la actividad de contactos será una nueva pantalla que contendrá los siguientes elementos y funcionalidades:

RF3.1.1.- Una caja de texto, donde el usuario podrá realizar una búsqueda caliente sobre el conjunto de tipos de metadatos existentes. Según se va escribiendo y se van filtrando los tipos de metadatos que coincidan, en el caso de no existir ninguno, al lado de la caja de texto se habilitará un icono de inserción para que el usuario cree ese nuevo tipo de metadato que está escribiendo.

RF3.1.2.- Una lista con el conjunto de tipos de metadatos ordenada de forma alfabética. Esta lista es del tipo 'elemento-detalle' al igual que el resto de listas del aplicativo, de modo que al pinchar sobre un tipo de metadato en particular se abrirá el detalle con el conjunto de valores que puede tomar ese tipo de metadato.

RF3.1.3.- El primer elemento de la lista será uno definido para la inserción de un nuevo tipo de metadato. Un tipo de metadato contendrá la información de nombre y descripción.

RF3.1.4.- Al pinchar prolongadamente sobre un tipo de metadato entraremos en un diálogo de modificación del tipo de metadato, pudiendo modificar sus propiedades de nombre y descripción. Desde este diálogo se permitirá eliminar el tipo de metadato.

RF3.1.5.- Al pinchar sobre un elemento de la lista de tipos de metadatos se nos abrirá su detalle con la lista de valores de tipo de metadato.

RF3.1.6.- La lista de valores por cada tipo de metadato tendrá un primer elemento que sirva para la inserción de un nuevo registro. Un valor de tipo de metadato contendrá la información de nombre y descripción (además del tipo de metadato al que pertenece)

RF3.1.7.- La lista de valores por cada tipo de metadato estará ordenada de forma alfabética.

RF3.1.8.- Al pinchar prolongadamente sobre el valor de metadato se nos abrirá un diálogo para la edición del mismo (nombre y descripción). Además, desde esta pantalla podremos eliminar el valor de metadato.

RF3.1.9.- Si pinchamos levemente sobre el valor del metadato se nos desplegará un diálogo con el conjunto de contactos ordenados de forma alfabética. Al lado de cada contacto habrá un check que nos permitirá indicar si ese contacto tiene o no el valor del metadato.

RF3.2.- Se hará una gestión de metadatos para los eventos similar al de los contactos.

RF3.3.- Se hará una gestión de metadatos para las notas similar al de los contactos y eventos.

RF4.- En la pantalla de contactos, al pinchar sobre un contacto se nos desplegarán igualmente sus datos de 'Teléfonos', 'Emails', 'Eventos' y 'Notas', aunque estos se agruparán en una pequeña tabla de dos filas por dos columnas. Tras estas dos filas; una nueva fila de tamaño variable tendrá listado de forma ordenada los tipos de metadato y metadatos que tenga asociado el contacto a modo lista informativa. Al

lado de esta lista un botón de detalle de metadatos. Tras esta fila, por último, estará de nuevo la fila de eliminación de contacto con el icono ya establecido.

RF4.1.- Al pulsar sobre el botón de detalle de metadatos se nos abrirá una nueva pantalla para la gestión de metadatos para un contacto en particular.

RF4.1.1.- La pantalla de gestión de metadatos de un contacto tendrá en la cabecera los datos del contacto. Seguidamente un combo con los tipos de metadatos para contactos y bajo este otro combo con los valores que puede tomar ese tipo de metadato o una caja de texto. Esto se definirá en el tipo de valor que pueda tomar el tipo de metadato; si es un valor enumerado o si es un texto libre.

RF4.1.2.- La pantalla tendrá un botón de insertar metadato. Cuando el usuario rellene un tipo de metadato y un valor (bien sea del combo o de la caja de texto) y pulse en dicho botón de inserción, se añadirá un nuevo elemento sobre la lista de metadatos que tendrá el contacto.

RF4.1.3.- La lista de metadatos que tiene el contacto tendrá por elemento un icono para su eliminación. Al pulsar sobre este icono desaparecerá el elemento sobre la lista.

RF5.- Al pulsar sobre el botón de búsqueda avanzada en el diálogo de opciones que había en el menú principal se nos abrirá la pantalla de búsqueda avanzada de contactos.

RF5.1.- La pantalla de búsqueda avanzada será similar a la de gestión de metadatos por contactos. Es decir, habrá un combo con los tipos, tras este otro combo o una caja de texto y tras ello una lista de valores de metadatos. Habrá también un botón de búsqueda que cerrará la pantalla de búsqueda avanzada y retornará a la pantalla de contactos teniendo un filtraje sobre los metadatos que se hayan introducido. Además, la cabecera de esta pantalla, donde está la caja de texto y el botón menú, cambiará a un color amarillo para señalar que hay filtro de metadatos.

RF6.- La gestión de metadatos y búsquedas avanzadas en la pantalla de eventos será similar al de la pantalla de contactos.

RF6.1.- El botón actual de añadir evento de la pantalla de eventos se sustituye por un botón menú. Al pulsar sobre este botón menú se abre un diálogo con las opciones de ‘añadir nuevo evento’, ‘Búsqueda avanzada’, ‘Gestión de metadatos’.

RF6.2.- Al entrar en el detalle de cada evento, el nº de contactos y nº de notas se unirá en una misma fila. Tras esta fila se incluye una nueva fila con los valores de metadatos que tuviese el evento. La última fila, como ya era antes, será para contener un botón de eliminación del evento.

RF7.- La gestión de metadatos y búsqueda avanzadas en la pantalla de notas será similar al de la pantalla de eventos y contactos.

RF7.1.- En la pantalla de notas, se cambiará el botón de inserción de nueva nota por un botón menú. Al pulsar sobre ese botón menú se abrirá un diálogo con la siguiente lista de opciones: 'Añadir nueva nota', 'Búsqueda avanzada de notas', 'Gestión de metadatos'.

RF7.2.- Al entrar en el detalle de cada nota, se juntarán el nº de contactos y el nº de eventos en la misma fila. Seguidamente, una nueva fila informará sobre los metadatos de la nota. Por último habrá una fila que permitirá la eliminación de la nota tal y como está en la actualidad.

RF8.- Para las notas se permitirá la inserción multilinea en el campo descripción.

RF9.- Las fechas deben contener no solo fecha sino también hora.

3.2.2.- Requisitos de Usabilidad:

RU1.- Las pantallas de contactos, eventos y notas, cuando tengan filtros activos, modificarán su cabecera poniéndola de color amarillo para indicar que hay filtros activos.

RU2.- La cabecera principal se suprime; ya no aparecerá en todo momento el icono de la aplicación con su nombre.

3.3.- Requisitos Fase II:

En este último apartado se definen los requisitos para establecer la comunicación entre terminales. Se define como se deben comunicar, que campos, como actuará la aplicación, etc.

3.3.1.- Requisitos Funcionales:

RF1.- Se incluirá en el detalle de cada contacto, evento y nota una fila de comunicaciones. Esta fila de comunicaciones contendrá una caja check que indicará si ese elemento se ha mandado a comunicar. En el momento que se indique que un elemento se quiere comunicar se abrirá un diálogo con el conjunto de contactos a comunicar. Se seleccionará un conjunto de contactos y se dará a un botón aceptar para que se envíe la información.

RF2.- Al pulsar sobre el icono de comunicaciones se podrá ver el histórico de cambios sobre el elemento. Es decir, si varios usuarios hacen cambios sobre un elemento, este tendrá la información de la última actualización. No obstante, se podrá ver un pequeño resumen de modificaciones que haya habido.

3.3.2.- Requisitos de Sistema:

RS1.- Los terminales deben ser capaz de comunicar constantemente sin hacer un uso excesivo de batería. Deben comunicar con servicios Host que resuelvan automáticamente las problemáticas de las comunicaciones.

RS2.- La información que se tendrá almacenada se guardará siempre en BBDD; no habrá ficheros o carpetas de almacenaje.

3.3.3.- Requisitos de Usabilidad:

RU1.- Las comunicaciones serán algo totalmente trasparente al usuario. El usuario no deberá configurar, parametrizar o tratar ningún ámbito relacionado con las comunicaciones. La única tarea del usuario es seleccionar que se desea la comunicación y con quien se desea dicha comunicación.

4.- Manual de uso:

4.1.- Resumen:

El presente manual explica el uso de la herramienta y el alcance de la misma. Mediante un conjunto de ilustraciones y unas sencillas indicaciones seremos capaces de realizar cualquier acción permitida en el aplicativo.

4.2.- Objetivos y alcance:

En este manual se explicará el siguiente conjunto de acciones:

- Búsqueda de un contacto mediante texto.
- Búsqueda de un contacto mediante voz.
- Importación de contactos mediante la agenda de teléfonos o LinkedIn.
- Unión de contactos en un solo registro.
- Añadir, modificar y eliminar un contacto.
- Añadir notas o eventos a un contacto.
- Crear, modificar y eliminar eventos.
- Buscar un evento en un periodo de fechas.
- Asignar contactos o notas a un evento.
- Añadir, modificar y eliminar notas.
- Buscar una nota por título o por fecha.
- Añadir contactos o eventos a notas.
- Realizar llamadas
- Envío de emails.

Estas son el conjunto de tareas que el presente manual trata de explicar.

4.3.- Destinatarios:

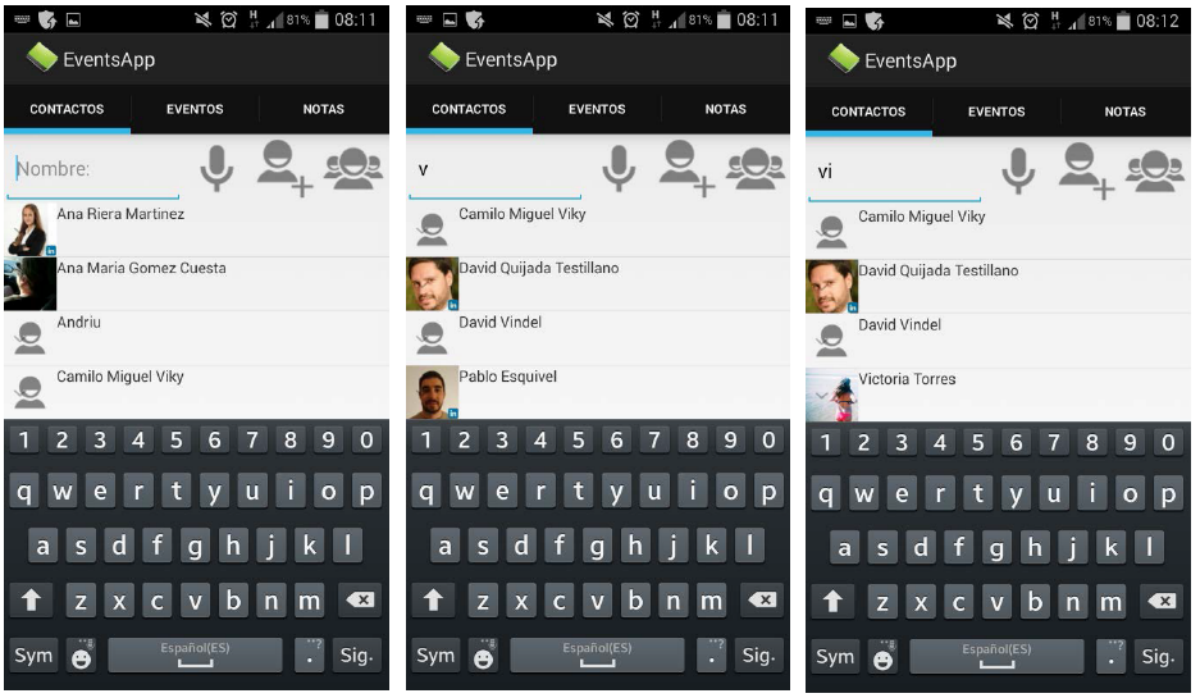
Este documento va dirigido a toda persona que necesite un uso avanzado del aplicativo y quiera tener un conocimiento perfecto del funcionamiento del mismo.

4.4.- Acciones:

4.4.1.- Búsqueda de un contacto por texto:

En esta acción se quiere localizar un contacto dado insertando texto. El texto debe ser nombre y/o apellidos en este orden. La búsqueda se hará en caliente (por cada letra introducida se irá realizando un filtro) y no distinguirá entre mayúsculas o minúsculas.

1.- Debemos posicionarnos en la pestaña 'CONTACTOS'. Desde esa pestaña, en la parte superior de la pantalla encontraremos una caja de inserción donde aparecerá un texto 'NOMBRE'. Debemos pinchar sobre esa caja y escribir el nombre a buscar. Veremos que la lista de contactos se va filtrando según vayamos escribiendo.



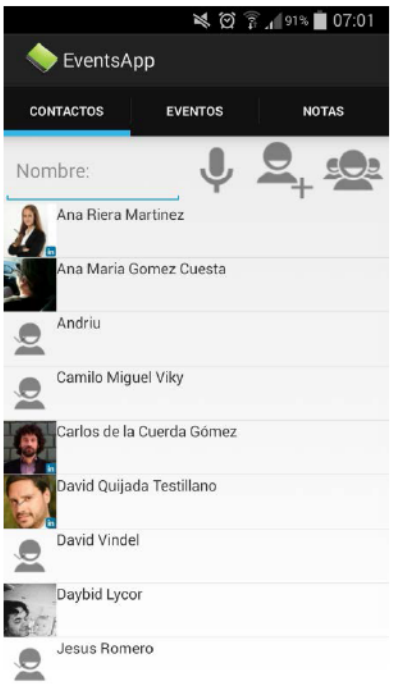
Aquí estamos buscando al contacto 'Victoria Torres'. Al principio nos aparece toda la lista de contactos pero vemos que según vamos escribiendo se nos va filtrando el conjunto de contactos hasta que finalmente lo visualizamos en pantalla.

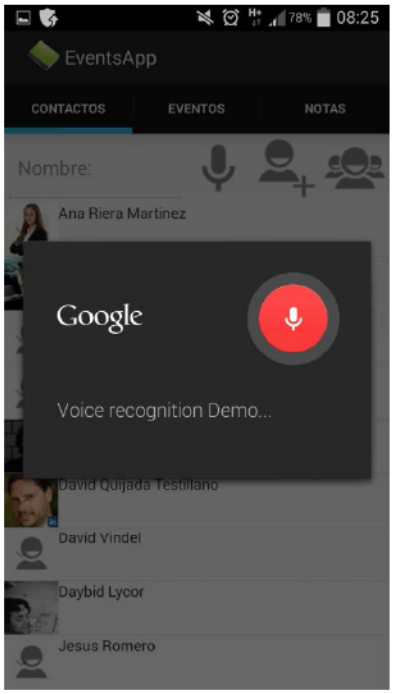
4.4.2.- Búsqueda de un contacto por voz:

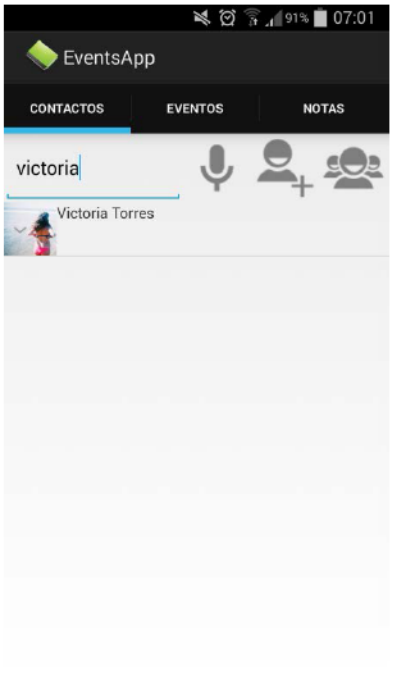
Este caso de uso trata de realizar una búsqueda sin la necesidad de escribir texto en el terminal. Puede ser útil para momentos en los que no podamos distraernos en ver lo que estemos escribiendo y solo podamos hacer indicaciones por voz.

1.- Nos posicionaremos en la pestaña 'CONTACTOS' y pulsaremos sobre el icono del micrófono.

2.- En ese momento, nos aparecerá una pantalla para inserción de voz. Hablaremos ahí y diremos el contacto que deseemos buscar.







De nuevo hemos buscado al contacto 'Victoria Torres'. Para ello, una vez nos ha aparecido la ventana de inserción de voz, hemos pronunciado 'Victoria' y se nos ha escrito automáticamente en la caja de texto, haciendo los filtros correspondientes.

Es importante notificar para esta funcionalidad la necesidad de internet puesto que es un servicio que ofrece Google en una de sus API.

4.4.3.- Importación de contactos mediante la agenda de teléfonos o LinkedIn:

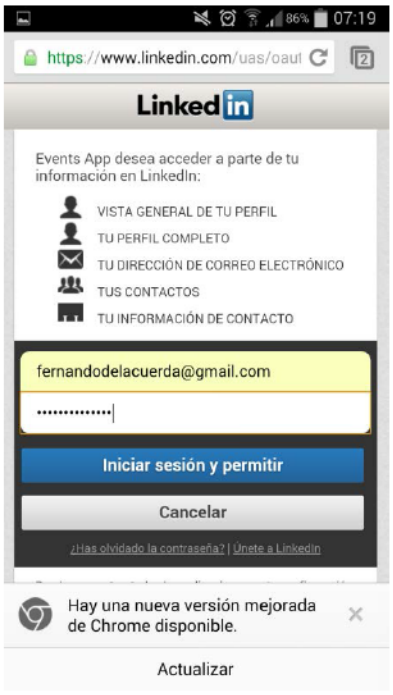
En este caso lo que se pretende es poder conectarse a la agenda de teléfonos o a LinkedIn y poder descargar la información que estos contengan.

Ambos casos de uso son similares por lo que los agruparemos en este apartado. Explicaremos la importación de datos de LinkedIn porque tiene una pantalla de Login que la agenda de teléfonos no requiere.

1.- Desde la pestaña de 'CONTACTOS' pulsamos el icono de 'Agrupación de contactos'. Es en el que aparecen un grupo de avatares.

2.- Seleccionamos la opción deseada, en este caso elegiremos la descarga de LinkedIn.

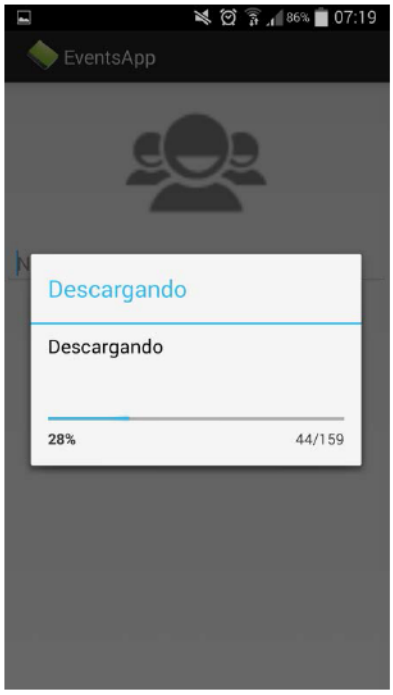
3*.- La aplicación nos lleva a la página de LinkedIn y nos requiere el usuario y password. Vemos también que nos pide permiso para acceder a determinada información de nuestros contactos. Introducimos nuestras claves y damos a 'Iniciar sesión y permitir'.

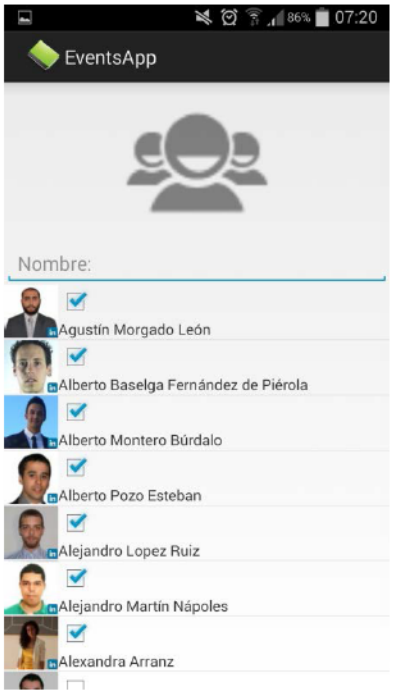


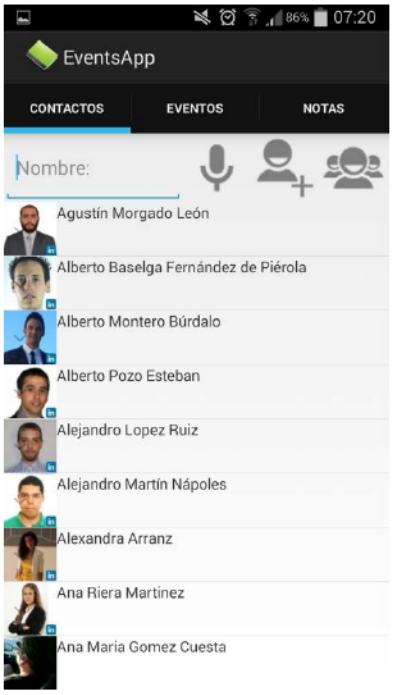
4.- Veremos que se abre un cuadro de diálogo en el que se nos muestra en tiempo real el estado de descarga de los contactos. Esperamos a que finalice.

5.- Nos aparece una lista de contactos con una caja para check. Activamos el check en todos aquellos contactos que queramos importar.

6.- Finalmente, pulsamos sobre el botón retroceder hasta volver a la pantalla inicial en la pestaña 'CONTACTOS'. Veremos que tenemos los contactos importados en nuestro listín.







Cómo podemos observar, tras importar los contactos de LinkedIn nuestra agenda a aumentado en número (antes empezaba por el contacto 'Ana Riera Martínez' mientras que ahora empieza por el contacto 'Agustín Morgado León'.

4.4.4.- Unión de contactos en un solo registro.

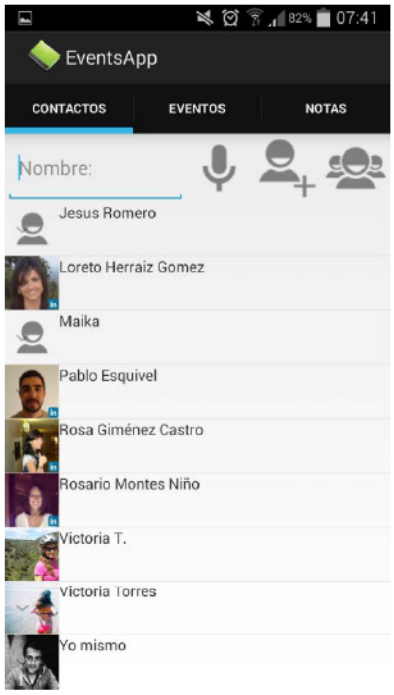
Se pretende unir varios contactos en uno solo. La información que de estos se tenga deberá por tanto combinarse para no perderse. Esta funcionalidad es debido a que la información la podemos importar de diferentes sitios y podemos requerir unificar contactos.

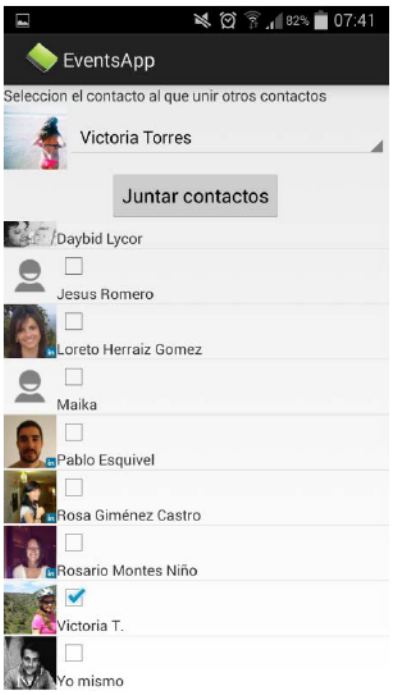
1.- Desde la pantalla principal, desde la pestaña 'CONTACTOS', seleccionamos el icono de 'Agrupación de contactos' (es aquel en el que aparecen varios avatares).

2.- Seleccionamos la opción de juntar contactos.

3.- En el desplegable de la parte superior seleccionamos el contacto padre. Es decir, el resultado tras unir varios contactos será un contacto con el nombre, apellidos y foto que aparezca en el contacto del desplegable. El resto de información se combinará con dicho contacto.

4.- Seleccionamos el contacto o contactos a unificar y pulsamos sobre 'Juntar Contactos'. Tras ello, pulsamos sobre la tecla retroceso para volver a la pantalla principal, pestaña 'CONTACTOS', donde veremos que los contactos se han unificado como hemos dictaminado.





En este caso hemos querido unificar los contactos 'Victoria Torres' con 'Victoria T.'. Al seleccionar 'Victoria Torres' como contacto padre, el contacto tras unificarse tendrá este nombre y su foto de perfil.

4.4.5.- Añadir, modificar, eliminar contactos, eventos y notas:

Lo que son las tareas básicas de datos sobre elementos se trabajan de igual forma en cada concepto: contactos, eventos y notas.

Dista más como realizar cada tipo de operación que el concepto sobre el que aplica, por lo diferenciaremos cada operación:

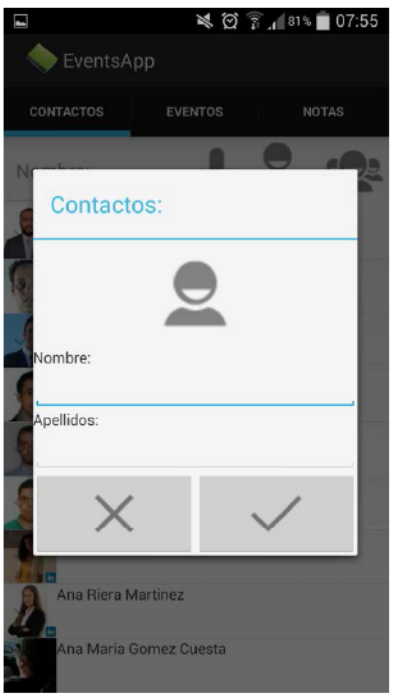
Añadir contactos, eventos y notas:

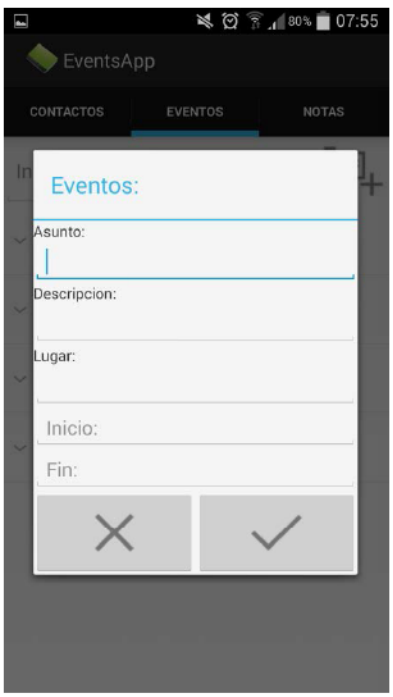
Se trata de crear, manualmente, un registro de cada tipo. Para ello deberemos realizar los siguientes pasos.

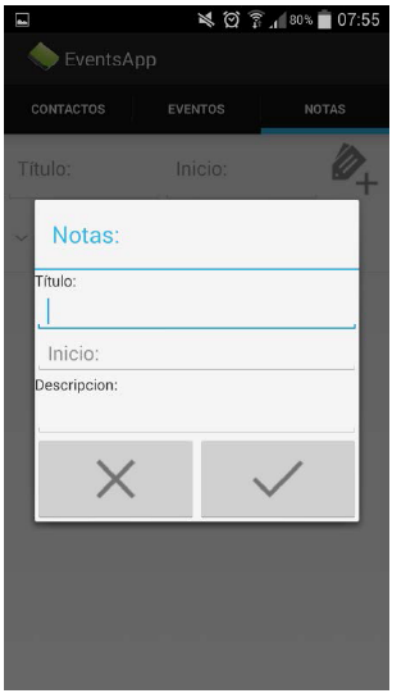
1.- Ir a la pestaña correspondiente y seleccionar el icono de inserción (aparecerá un contacto, nota o evento junto con un aspa que nos indique que se añade).

2.- Rellenar los datos a introducir. En caso de que nos falte algún dato, en el momento que demos sobre el botón aceptar nos aparecerá un mensaje por pantalla indicando que faltan algunos campos obligatorios.

En el caso de los contactos, si pulsamos sobre el icono nos llevará a un gestor de imágenes para que seleccionemos una imagen del contacto (esta funcionalidad se extenderá a los eventos y notas en el Evolutivo I).







Cómo podemos observar, los tres diálogos son muy parecidos. Simplemente cambia el contenido de la información a guardar.

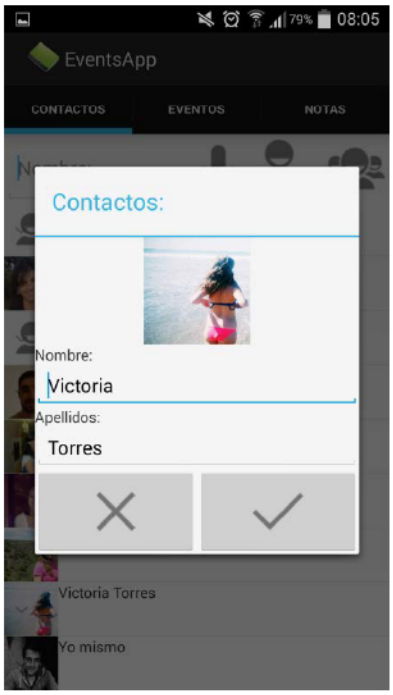
Modificar contactos, eventos y notas:

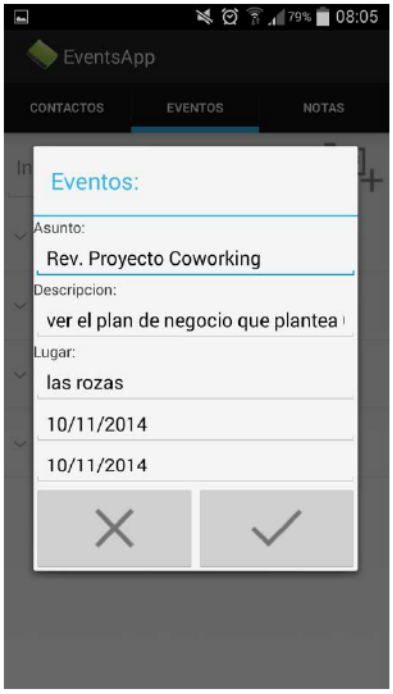
Ahora lo que necesitamos es modificar algún registro en concreto.

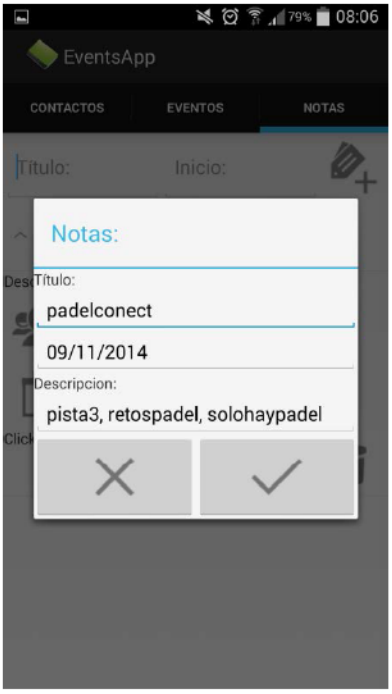
1.- Desde la pantalla principal ir a la pestaña correspondiente. Pulsar y mantener pulsado el registro que queremos editar (deberemos pulsar alrededor de 1-1,5 segundos).

2.- Se nos abrirá un diálogo similar al de 'inserción de nuevo' registro pero con los datos rellenos. Deberemos modificar los campos que queramos (en el caso de contactos, también podremos editar la foto).

3.- Pulsaremos sobre el OK para guardar los cambios.







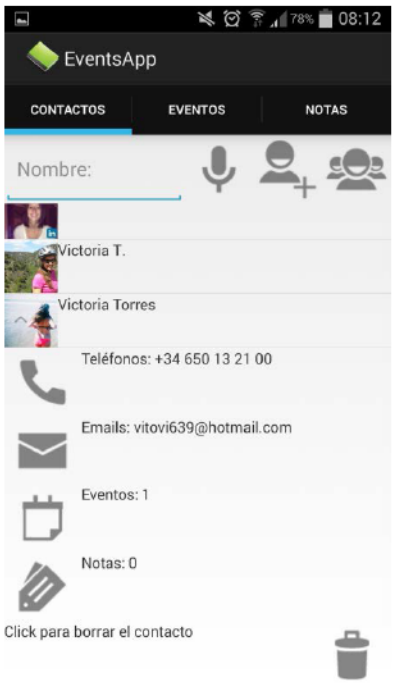
Eliminar contactos, eventos y notas:

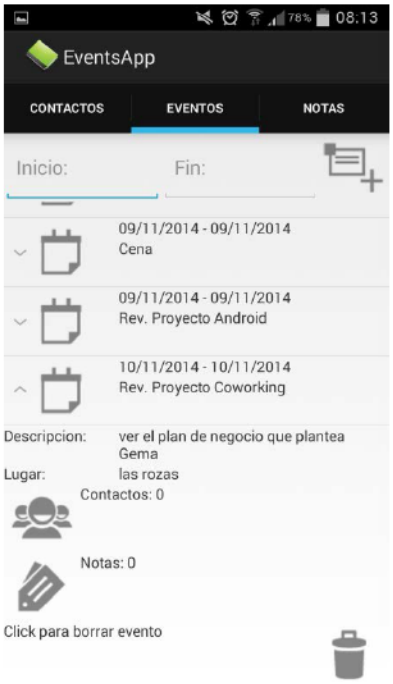
Ahora detallaremos como eliminar un registro.

1.- Deberemos ir a la pantalla principal desde la pestaña correspondiente, pinchar levemente sobre el registro a eliminar.

2.- El registro se nos desplegará en detalle, aportándonos información del mismo y mostrándonos un icono de papelera para que podamos eliminarlo.

3.- Pulsamos sobre el icono papelera y aceptamos la confirmación que nos aparece.





En resumen podemos ver como las tres operaciones básicas sobre datos (creación, modificación y eliminación) se tratan de igual forma para cada concepto.

4.4.7.- Relación entre Contactos, Eventos y Notas:

A continuación se muestra cómo podemos relacionar cada uno de los conceptos básicos del aplicativo (contactos, eventos y notas).

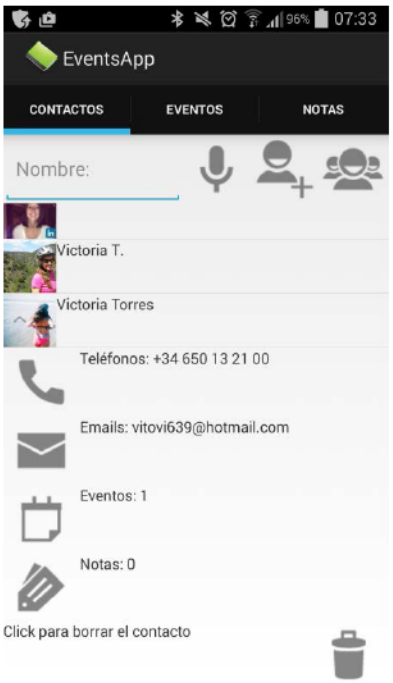
El tipo de relaciones que pueden alcanzar estos elementos son de NxM:

- Un contacto puede tener N eventos asociados.
- Un contacto puede tener N notas asociadas.
- Un evento puede tener N contactos asociados.
- Un evento puede tener N notas asociadas.
- Una nota puede tener N contactos asociados.
- Una nota puede tener N eventos asociados.

Para poder crear estas relaciones en el aplicativo deberemos:

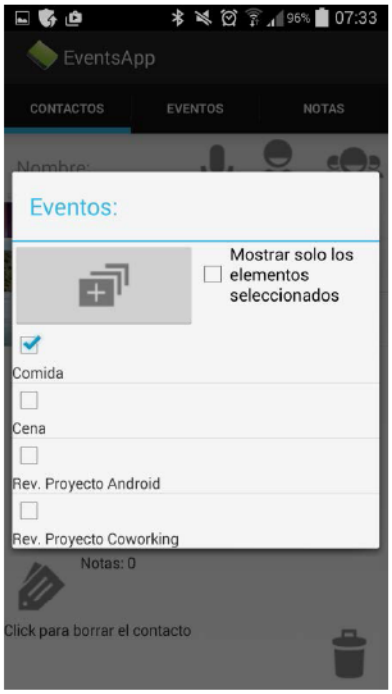
1.- Ir a la pestaña del elemento que queramos relacionar. Si queremos relacionar por ejemplo un contacto con un evento, lo podremos hacer tanto de la pestaña contactos como de la pestaña eventos.

2.- Pinchar brevemente sobre el ítem a relacionar. Se nos abrirá el detalle del ítem.



3.- Pincharemos sobre la definición del elemento al que queremos relacionarnos. En caso de que pinchemos sobre su icono en vez de sobre su definición, nos aparecerá el cuadro de diálogo para crear un nuevo registro de ese tipo. Es decir, nos ayuda a crear un elemento dado sin tener que ir a la pestaña correspondiente y crearlo desde allí.

4.- Seleccionamos el conjunto de elementos a los que nos queremos relacionar. Podemos filtrar este subconjunto activando el check superior para ver los elementos relacionados.



Cómo podemos observar, en la pantalla anterior se nos estaba indicando que el contacto Victoria Torres estaba asociada a un evento. Al abrir el listado de eventos para asociar con Victoria Torres vemos que nos aparece un evento seleccionado.

4.4.8.- Realización de llamadas o envío de emails:

Se agrupan estas funcionalidades puesto que su definición y uso es muy similar. Se trata poder realizar llamadas y enviar emails desde el aplicativo.

Los contactos tienen la peculiaridad de poder estar asociados a N teléfonos y M emails. Para poder realizar estas acciones:

1.- Desde la pestaña contactos deberemos seleccionar al contacto con el que queramos comunicarnos pinchando levemente sobre este.

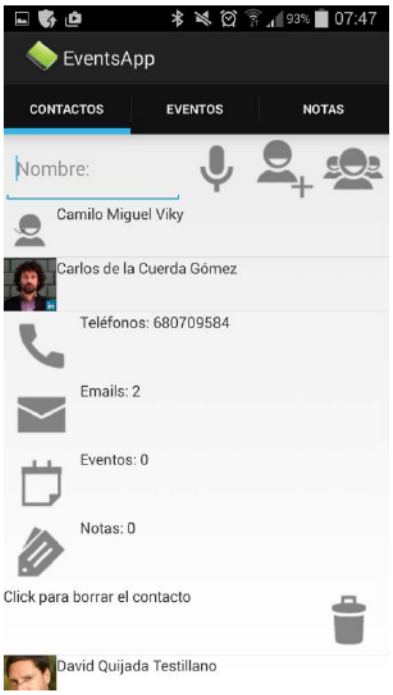
2.- Se nos desplegará la pantalla detalle del mismo donde veremos 2 iconos; uno para realizar llamadas y otro para el envío de emails. Pinchamos sobre este:

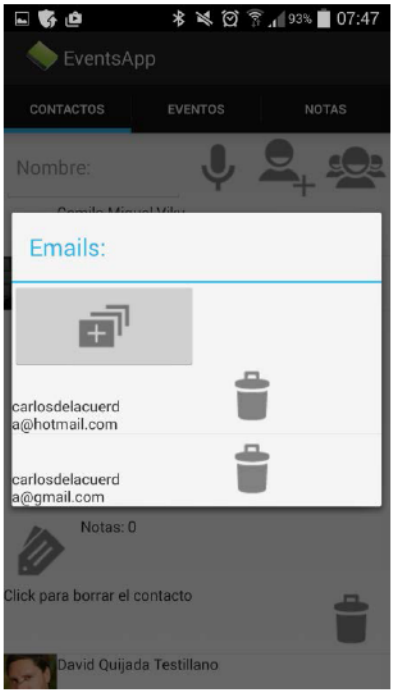
- En caso de no tener ningún registro, nos aparecerá un diálogo para que insertemos un nuevo registro

- En caso de tener un registro, nos llevará automáticamente al controlador de llamadas o gestor de emails para que enviemos un email.

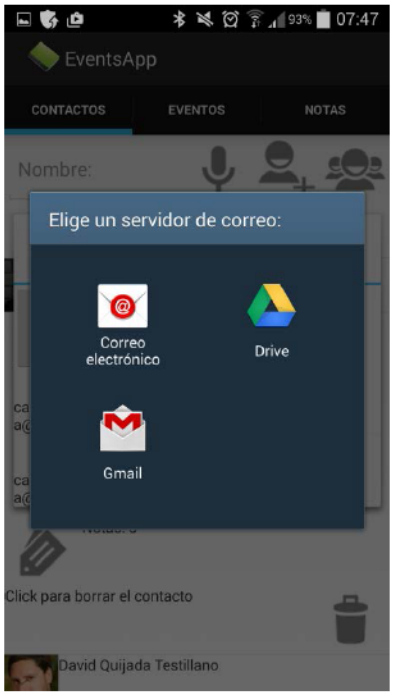
- En caso de tener más de un registro nos llevará a un diálogo de selección para que especifiquemos el registro a tratar. También podremos crear nuevos registros.

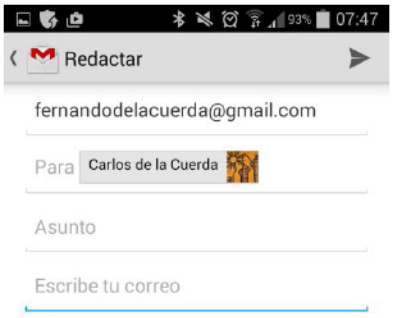
Si queremos crear, modificar, eliminar o simplemente visualizar registros que hay, deberemos pulsar sobre el texto de definición y no sobre el icono.





3.- Dependiendo de si hemos pulsado sobre el envío de emails o realización de llamadas, se nos abrirá un componente u otro referente a cada ámbito.





5.- Desarrollo:

Este apartado habla sobre los fundamentos realizados en la parte de desarrollo. Explicaremos aspectos como la estructura de base de datos, patrones de diseño, arquitectura del proyecto, metodologías de programación, etc.

A la hora de afrontar un proyecto software debemos hacer un estudio detallado no solo de lo que vamos a hacer sino de cómo vamos a hacerlo.

Seguir una política determinada en el desarrollo permite que el proyecto esté mejor organizado, teniendo de este modo un mantenimiento más sencillo y la virtud de no depender de los conocimientos del programador que lo implementó.

5.1.- Base de Datos:

Para la Fase I, que es totalmente móvil, requeríamos de una base de datos donde alojar nuestra información ya que los datos no debían ser volátiles.

5.1.1.- SQLite:

El Sistema Operativo Android contiene por defecto la posibilidad de crear bases de datos SQLite en sus terminales, por lo que hemos optado por esta herramienta para el manejo de nuestra información.

SQLite es un tipo de base de datos relacional contenida en una pequeña biblioteca escrita en C. Es de carácter libre y basa su esquema de tratamiento en el estándar SQL. Además, se integra perfectamente en los aplicativos Android y el tamaño máximo capaz de albergar es suficiente para las necesidades del proyecto (2TB).

A diferencia de otras bases de datos como SQL Server, SQL Azure u Oracle, SQLite permite la inserción de campos mal tipados. Es decir, podríamos insertar un registro STRING en un campo tipo NUMBER. Este hecho puede provocar lógicamente inconsistencias en los datos, por lo que los aplicativos que utilicen la herramienta deben de tener la certeza de tratar estas inconsistencias.

También podemos encontrar otras erratas como la posibilidad de introducir campos nulos en las claves primarias o registros duplicados.

En definitiva, SQLite es una base de datos relacional, basada en una arquitectura SQL pero que no tiene los métodos básicos de seguridad de datos que suelen tener este tipo de Base de Datos como la validación de datos y tipos de datos. No obstante, se integra con facilidad en terminales móviles y sus tiempos de respuesta y capacidades son más que ventajosas.

Nuestro aplicativo, al tratar con este tipo de base de datos, debe tener unas clases de acceso robustas y atómicas que permitan tener la certeza de no corromper los datos.

5.1.2.- Esquema de Base de Datos:

En el siguiente esquema de BBDD podemos visualizar el conjunto de tablas que componen nuestro sistema.

A la hora de realizar el diseño se podría haber optado por múltiples opciones, pero se ha evaluado cuales serían las más sencillas de implementar y mantener en un futuro.

Tenemos tres tablas principales que contienen la información básica de los elementos 'Contacto', 'Nota' y 'Evento'. Hemos creado por tanto una tabla para cada concepto cuando podríamos haber creado una tabla común y de ella haber añadido propiedades para cada caso en particular.

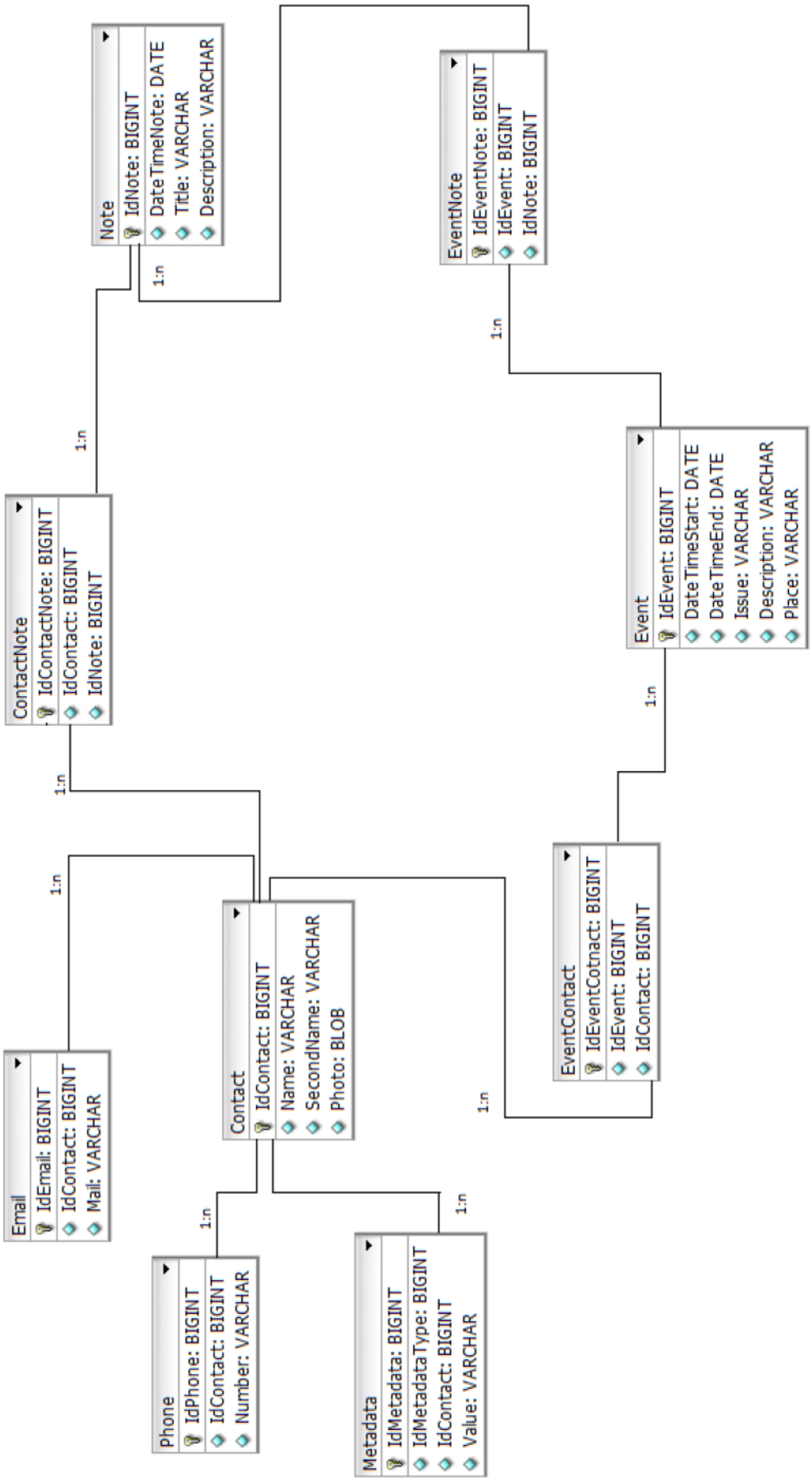
Por otro lado, algo que también llama la atención es que en las tablas de relación encontramos un ID que no es la combinación de las tablas relacionadas sino que es un ID propio. Esto se ha hecho así por varios motivos:

- Para la Fase II el sistema de comunicaciones se plantea como un servicio Web que vaya asignando claves a los diferentes registros bajo petición. Esto requeriría que cada tabla tuviese un único identificador único.
- Para el Evolutivo I, Fase II o posibles evolutivos o fases posteriores, se puede requerir la necesidad de añadir información en las relaciones. Por ejemplo, cuando se hizo la relación, quién la hizo, por qué la hizo, etc. Dejando una sola clave como identificadora los cambios a realizar en múltiples consultas y clases son de menor tamaño.

También destaca la existencia de una tabla llamada 'Metadata'. Aunque se ha hablado que el tratamiento de metadatos es del Evolutivo I, en esta primera fase era necesario introducir metadatos para indicar la fuente de origen de los contactos; es decir, si provenían de LinkedIn o de la Agenda de teléfono.

Vemos por esta línea la falta de una tabla 'MetadataType'. Esto es porque aunque se hace referencia a un identificador, este identificador es estático puesto que esta tabla solo la trabaja internamente la aplicación, por lo que los registros de tipo son estáticos (1 para la agenda, 2 para LinkedIn).

En un futuro, cuando se incorpore el tratamiento de metadatos y tipos de metadatos, habrá que tener cuidado con los tipos de metadatos para que no se pisen unos a otros.


Note
IdNote: BIGINT
DateTimeNote: DATE
Title: VARCHAR
Description: VARCHAR
EventNote
IdEventNote: BIGINT
IdEvent: BIGINT
IdNote: BIGINT
Event
IdEvent: BIGINT
DateTimeStart: DATE
DateTimeEnd: DATE
Issue: VARCHAR
Description: VARCHAR
Place: VARCHAR
ContactNote
IdContactNote: BIGINT
IdContact: BIGINT
IdNote: BIGINT
Contact
IdContact: BIGINT
Name: VARCHAR
SecondName: VARCHAR
Photo: BLOB
EventContact
IdEventCotnact: BIGINT
IdEvent: BIGINT
IdContact: BIGINT
Email
IdEmail: BIGINT
IdContact: BIGINT
Mail: VARCHAR
Phone
IdPhone: BIGINT
IdContact: BIGINT
Number: VARCHAR
Metadata
IdMetadata: BIGINT
IdMetadataType: BIGINT
IdContact: BIGINT
Value: VARCHAR
1:n

5.2.- Arquitectura del proyecto.

Para este proyecto se ha utilizado una arquitectura sencilla, orientada a objetos y separada por capas.

El proyecto se compone por un total de 48 clases de negocio y múltiples ficheros de recursos, idiomas e imágenes.

- 20 clases de apoyo a pantallas.
- 11 clases manejadoras de datos.
- 5 clases de utilidades.
- 11 clases de adaptadores.
- 24 ficheros xml de diseño.
- Múltiples ficheros de imagen
- Ficheros de recursos (textos y colores).
- Ficheros de idiomas.
- Ficheros de estilos, dimensiones, fondos, etc.

A la hora de plantearnos la arquitectura del proyecto debemos pensar ¿Cuál es el alcance de nuestro proyecto? ¿Qué posibles evoluciones va a tener? ¿En qué tipo de plataforma se va a desarrollar? ¿Qué tipo de desarrollador va a implementarlo? Etc.

Sobre el papel siempre se busca optimizar el código y hacerlo lo más escalable posible, pero, en la realidad, hay que tener muy presente el ámbito del proyecto en el que se está trabajando para saber si debemos llevar a cabo todo tipo de metodología o regla de diseño pues esto podría ser contraproducente en muchos sentidos.

Pongamos varios ejemplos para tratar de explicar esta teoría; en nuestra aplicación tenemos 3 tablas (Contacto, Nota y Evento) que hacen referencia a los elementos principales del aplicativo. Podríamos haber optado por la creación de una sola tabla (ElementoBásico) con los campos comunes a las 3 y luego, por su lado, una tabla específica para cada elemento. Esto en nuestro caso nos habría generado 4 tablas (la principal genérica más una específica por cada elemento básico). Además, habría un campo duplicado que en el primer esquema no está y es el campo ID, para relacionar la tabla principal con la tabla específica.

Parece entonces que el primer esquema es más óptimo pero, si vamos un paso más allá vemos que podría no ser así; Ahora queremos relacionar los elementos entre sí. En nuestro esquema hemos creado 3 tablas de relaciones; ContactoNota, EventoNota,

ContactoEvento. Sin embargo, para el esquema propuesto con una sola tabla nos valdría: ElementoElemento. Es más, si hacemos una pequeña tabla podemos ver como el esquema original se queda muy atrás según vayan creciendo los elementos básicos:

Nº de tipos elemento	Nº tablas esquema actual	Nº tablas esquema teórico
1	-	-
2	1	1
3	3	1
4	6	1
5	10	1
6	15	1
7	21	1
8	28	1
9	36	1
10	45	1
…	…	…
100	5050	1

Viendo esta tabla parece que no debería haber duda a la hora de elegir un esquema u otro y, sin embargo, hemos optado por el peor esquema. Estos son, entre otros, los motivos por los que hemos tomado esta decisión:

- En nuestro proyecto se ha determinado que iba a tener un máximo de 3 elementos básicos. Además, en un futuro, uno de estos elementos básicos pasará a ser un metadato por lo que el esquema se quedará con 2 elementos básicos.
- La creación de tablas separadas nos obligará a realizar consultas algo más complejas (mediante intersecciones de las mismas) y mayor número de 'ataques' a la base de datos.
- El desarrollador que tome el proyecto deberá tener un conocimiento previo sobre el esquema ya que este no es tan intuitivo.
- Para la Fase II, con el tratamiento de comunicaciones, al estar tablas separadas la lógica se complica (mayor información a enviar). Además, podría haber problemas con claves foráneas que en el esquema básico no se dan (imaginemos que enviamos la información al servidor de un elemento básico y su especificación y que solo llega esta última → error en BBDD por clave ajena no encontrada).
- En caso de haber tenido una mala concepción del proyecto a futuro, el esquema básico inicial no limita que en un futuro se cree el otro esquema asociado.

Pongamos ahora otro ejemplo que nos separe de la línea marcada por la teoría a la línea de la utilidad personalizada real; nuestras tablas de relaciones.

Por lo general, una tabla de relaciones suele tener una clave primaria formada por las claves primarias de cada elemento que hace relación. En caso de que pueda haber varias relaciones iguales en base a algún dato como fecha o metadato, entonces se incluye este identificador en la clave primaria. De este modo se consigue tener una tabla robusta y atómica, con las medidas de seguridad que ofrecen las claves primarias en las bases de datos.

Ahora, si nos vamos a nuestro proyecto, vemos que las tablas de relaciones (como ContactoEvento, EventoNota o ContactoNota) tienen un identificador propio, por lo que por un lado nos estamos saltando el patrón de diseño genérico en este tipo de tablas y, por otro, estamos almacenando un identificador cuando realmente no aporta aparentemente ninguna información añadida.

Para explicar esta decisión tenemos que enfocar la vista a la Fase II del proyecto; cuando vayamos a las comunicaciones y pensemos en detalle el esquema a seguir y, dado una experiencia ya adquirida en otros proyectos, el uso de una clave primaria única por cada tabla era muy atractivo a la hora de poder hacer las funcionalidades necesarias para la realización de comunicaciones.

Es decir, podemos montar un servicio web con una única función de recogida de datos del tipo 'GetDate(String tableName, Strign namePrimaryKey)'. Esta función sería usable por cualquier tabla de nuestro esquema y, sabiendo, que para futuras tablas también será válido.

Otro ejemplo en nuestro proyecto que difiere de lo establecido por las normas de diseño es el uso compartido de objetos de datos y modelos.

En un proyecto de gran alcance hay una separación lógica entre un modelo y un DAO. Esta separación se hace porque mientras el modelo es un objeto lógico que contiene el conjunto de propiedades y métodos para una pantalla, clase o conjunto de elementos, un DAO simplemente interpreta una tabla o vista de una base de datos y ofrece un conjunto limitado y cerrado de funcionalidades sobre dicha tabla o vista.

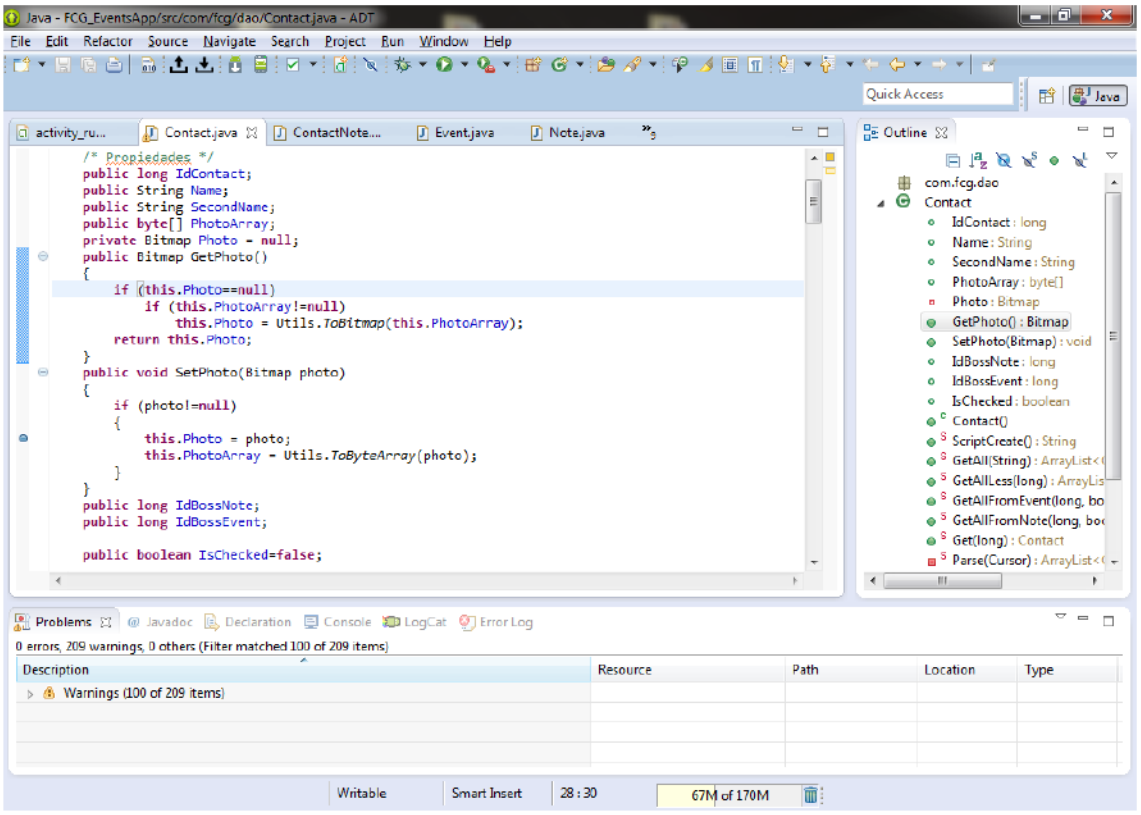
La separación de los modelos y los DAO es porque la parte de negocio debería trabajar únicamente con modelos y permitir de este modo la encapsulación de los datos. Al

modelo no debería afectarle que se trabajase con una base de datos, un sistema de archivos, servicios en la nube o cualquier otro origen de datos.

Los DAO sin embargo sí que van ligados al sistema de almacenamiento que se utilice y estos deberían ofrecer un API común al exterior pero implementar por dentro las características propias del sistema que trabajen.

En nuestro proyecto sin embargo se han combinado los modelos con los DAO para ofrecer un menor número de clases.

Al tratarse de un proyecto pequeño lo que se ha hecho es expandir la funcionalidad y características de los DAO para que se adaptasen a las necesidades de negocio. Por ejemplo, veamos la clase Contact:



En esta clase encontramos todas las propiedades referentes a la tabla de Base de Datos que son: *IdContact, Name, SecondName, PhotoArray*. Estos campos son con los que trabajamos en la base de datos. Sin embargo, nos encontramos con otro como *Photo* unido a sus método get y set (*GetPhoto, SetPhoto)*.

La propiedad *Photo* almacena en una variable lo que hay en el campo de BBDD *PhotoArray* pero transformado en un tipo legible por la aplicación (de array de bytes a

Bitmap). Esta funcionalidad no es del DAO, sino que estrictamente sería del modelo. Al igual pasa con la propiedad *IsChecked.* Esta propiedad hace referencia a una necesidad del negocio y no contempla su información en BBDD, pero se incluye en esta clase para poder tener toda la información agrupada.

Esta forma de trabajar, a gran escala podría causar grandes problemas dado que la clase crecería exponencialmente según se fuese necesitando información añadida como hemos visto con los casos de *GetPhoto, SetPhoto ó IsChecked.* No obstante, tras haber estudiado el proyecto y ver su alcance, hemos optado por esta arquitectura ya que en caso contrario, deberíamos haber creado una doble clase por cada DAO (el DAO y el modelo) además de otros modelos que no se relacionan con la BBDD. Este hecho habría incrementado notablemente el tamaño del proyecto y se ha desechado la propuesta.

En definitiva, a la hora de llevar esta metodología a la práctica nos encontramos ante un proyecto Android de dimensiones pequeñas, que ciertamente se espera que crezca pero que nunca va a ser un desarrollo especialmente amplio. Debemos tener en cuenta este tipo de hechos para poder llevar a cabo una arquitectura lógica.

Es decir, a la hora de hacer el diseño de arquitectura del mismo evaluamos el contexto de desarrollo del proyecto y nos encontramos con un solo programador para un proyecto de dimensiones pequeñas. Además, el proyecto cuenta con evolutivos pero de pequeño tamaño y no se prevé proyectos paralelos que puedan reutilizar parte del código.

Ahora bien, el proyecto va a sufrir evoluciones que no tienen por qué ser del mismo programador ni se han fechado aún. Además, el proyecto basa sus datos en una BBDD SQlite por lo que requeriremos una fuerte seguridad del aplicativo en el acceso a datos.

En resumen, podemos analizar estos dos últimos párrafos para determinar porqué utilizar unas metodologías u otras.

- Un solo programador: Esto implica que no tengamos la necesidad de fragmentar la solución en varios proyectos dado que no se va a trabajar en paralelo.
- Es un proyecto Android de dimensiones pequeñas del que no se espera un crecimiento notable: Quiere decir que no debemos realizar componentes

genéricos de los que hereden componentes específicos para su uso particular o que el propio código pueda estar interrelacionado y saltarse algunas normas de atomicidad. Efectivamente, sale código de menor calidad pero el tiempo de realización es mucho menor.

- No se prevé proyectos paralelos que puedan reutilizar parte del código: Significa que no debemos gastar tiempo en crear proyectos separados o clases demasiado genéricas ya que el código que aquí se genere va a ser sólo destinado a este proyecto.
- El proyecto va a sufrir evolutivos que no tienen por qué ser del mismo programador ni se han fechado aún: Este motivo implica que el código debe seguir una serie de patrones de desarrollo comunes tales como nombramiento de variables, uso de objetos, independencia del programador, etc. Debe ser un código legible para otros desarrolladores o para el mismo desarrollador ya que no se va a tratar la Fase II hasta un tiempo determinado.
- El proyecto basa sus datos en una BBDD SQlite por lo que requeriremos una fuerte seguridad del aplicativo en el acceso a datos: Aunque todos los proyectos deberían tener su parte de seguridad en datos desde el aplicativo, el hecho de que SQLite permita la creación de registros con claves primarias nulas, datos con tipos erróneos, duplicidades, eliminación de registros con claves ajenas, etc. Hace que el proyecto deba estar más pendiente de cómo trabaja los datos para no crear inconsistencias en la base de datos. De este modo, se ha creado por un lado un manejador personalizado de la base de datos (la clase BDManager.java). Por otro lado, se han creado objetos de datos por cada tabla (Contact.java, Note.java, Event.java, ContactNote.java, etc). Cada una de estas clases contiene los accesos a datos, sin ser accesible desde ninguna otra parte del código, por lo que se focaliza perfectamente todo el tema de datos.

5.2.3.- Programación por capas:

El proyecto contiene 3 capas diferenciadas; por un lado tenemos la capa de presentación, después la capa de negocio y por último la cada de datos. Estas capas son independientes entre sí y permiten hacer un desarrollo organizado y óptimo.

Además, la programación por capas permite el trabajo en paralelo o la limitación de responsabilidades. Este mismo proyecto podría haberse fragmentado en varios proyectos independientes y atómicos. Por ejemplo, todo el acceso a datos podría haber sido un proyecto a modo de 'API'. Otro proyecto podría haber llevado la lógica

de negocio consultando a este API y, por último, un tercer proyecto podría haber sido la implementación de interfaces gráficas.

Por ejemplo, para el Evolutivo I se plantea un lavado de cara del aplicativo. Esto requiere cambiar el diseño de los ficheros de layout e imágenes. Sin atendemos a esta necesidad, no necesitaríamos implementar nada de código y se podría subcontratar.

Si nos vamos al proyecto en particular, vemos que tenemos un Package para los DAO (donde ya hemos visto que pueden tener funcionalidades añadidas), otro Package para las actividades o pantallas (parte de negocio) y por último un directorio con los ficheros de diseño (res\layout).

El resto de carpetas o directorios son funcionalidades añadidas. También vemos capas intermedias a las principales como el Package de adaptadores, donde es un paso entre negocio y diseño.

La composición del proyecto por bloques diferenciados permite al desarrollador saber donde tiene que implementar nuevas funcionalidades, buscar focos de error y en definitiva, hacerse con el proyecto.

5.2.4.- Programación orientada a objetos:

Una programación orientada a objetos permite una programación muy potente. Los objetos nos entidades que tienen 3 características muy interesantes:

- Estado: Hacen referencia al conjunto de propiedades o características que contiene un objeto dado.
- Comportamiento: Es el conjunto de métodos que el objeto puede realizar y cómo va a realizarlos en base a su estado.
- Identidad: Es aquella característica o propiedad que lo identifica del resto de objetos de forma unívoca.

El proyecto *EventsApp* se basa en una programación Orientada a Objetos por su potencia y su expansión en el ámbito del desarrollo informático.

El caso más notable en el proyecto del uso de objetos son los DAO. Estos objetos representan registros de la Base de Datos. El desarrollador, mediantes esta

colección de objetos puede interactuar con la base de datos teniendo en todo momento la información agrupada relevante a un registro.

La encapsulación de la información en objetos permite que el acceso a datos se limite a estas clases DAO, por lo que tendremos la seguridad de que por muchas modificaciones que sufra el proyecto, sólo tendremos que comprobar su acceso a datos en este conjunto de objetos.

Hay varias teorías sobre la programación orientada a objetos; los más puristas llevan esta metodología al extremo eliminando cualquier método estático en sus desarrollos para trabajar siempre con instancias de objetos.

En este proyecto no se busca alcanzar esos límites y se combina el uso de métodos estáticos con el de métodos instanciados en base a la índole de este. Se sigue un patrón basado en unas reglas:

Métodos estáticos: serán aquellos métodos cuyo resultado no dependa del estado de ninguna propiedad o que actúe sobre un conjunto de elementos. Por ejemplo:

Public static ArrayList<Contact> GetAllContacts();

Este método recoge todos los contactos de BBDD y los devuelve en una colección de objetos de tipo 'Contact'. El resultado de este método no depende de ninguna propiedad y devuelve una colección.

También podemos confundirnos con otros métodos del tipo:

Public static Contact GetContact(long idContact);

Este método, también estático, tiene un atributo que influye sobre el resultado y devolve un solo registro. Sin embargo, este método es estático puesto que el objeto no se ha creado aún.

Los métodos estáticos como vemos pueden trabajar con objetos pero no sobre ellos.

Métodos instanciados (objetos): son aquellos métodos que trabajan sobre los objetos. Sus funcionalidades se basan en el estado del objeto. Por ejemplo:

Public bool Delete();

Este método borra una instancia del objeto y devuelve un valor boolean en base a su resultado (si ha habido errores).

Otro método pero con parámetros podría ser:

Public bool SetName(String newName);

En este caso, lo que está ocurriendo es que se está cambiando una propiedad del objeto, por lo que aparentemente parece lógico que sea un método instanciado.

Todos los métodos pueden pasar a ser estáticos cuando son instanciados y viceversa. El uso o forma de trabajar depende en numerosas ocasiones del desarrollador.

EventsApp trata de seguir una política permanente en su proyecto en este sentido y trabaja conjuntamente con unos métodos u otros según la tipología del problema.

En el siguiente fragmento de código vamos a poder ver la dinámica que se ha llevado durante la implementación de *EventsApp* y la compararemos con el extremismo en objetos y el extremismo en método estáticos.

El problema requiere sacar todos los contactos de BBDD cuyo nombre contenga la letra 'A'. Sobre la colección resultante se quiere modificar el apellido y poner 'Gómez'.

5.2.4.1.- Implementación EventsApp:

```
Public void Problema()
{
        //Método estático
        ArrayList<Contact> contactsOb = Contact.GetAll();
        Foreach(Contact contactOb: contactsOb)
        {
                //Método instanciado
                If (contactOb.Contains("A"))
                {
                        contactOb.SecondName = "Gómez";
                        //Método instanciado
                        contactOb.Update();
                }
        }
}
```

5.2.4.2.- Implementación métodos estáticos (con uso de objetos):

```
Public void Problema()
{
        //Método estático
        ArrayList<Contact> contactsOb = Contact.GetAll();
        Foreach(Contact contactOb: contactsOb)
        {
                //Método estático
                If (String.Contains(contactOb.Name, "A"))
                {
                        //Método estático
                        Contact.UpdateSecondName(contactOb.IdContact,
"Gómez");
                }
        }
}
```

5.2.4.3.- Implementación con instancias

```
Public void Problema()
{
        //Creación de objeto
        ContactDB contactDB = new ContactDB();
        //Método instanciado
        ArrayList<Contact> contactsOb = contactDB.GetAllContact();
        Foreach(Contact contactOb: contactsOb)
        {
                //Método instanciado
                If (contactOb.Contains("A"))
                {
                        contactOb.SecondName = "Gómez";
                        //Método instanciado
                        contactOb.Update();
                }
        }
}
```

Estos tres ejemplos realizan la msima funcionalidad y cómo vemos, el tamaño de su código no difiere notablemente. El uso de unas formas u otras depende más del desarrollador.

EventsApp trabaja con una programación híbrida porque consideramos que el uso tanto de métodos estáticos o instanciados puede ser más adecuado o no dependiendo de la problemática que se plantee para un caso dado.

5.3.- Presupuesto:

5.3.1.- Desglose de conceptos:

Concepto	***Cantidad (h)***	***Precio €***	***Total €***
Reuniones Iniciales resumen necesidad	16	30	720
Estudio entorno y viabilidad proyecto	24	30	1080
Creación de demo	24	30	1080
Aceptación y toma de requisitos	16	30	720
Documentación de requisitos	24	30	1080
Creación plantilla app con contenido visual	96	30	4320
Usabilidad simple entre pantallas	128	30	5760
Control de seguimiento del cliente	8	30	360
Pantalla de contactos	64	30	2880
Pantalla de eventos	56	30	2520
Pantalla de notas	56	30	2520
Funciones de relación entre elementos	24	30	1080
Acceso API agenda	24	30	1080
Acceso API LinkedIn	32	30	1440
Gestión de idiomas	16	30	720
Clase de acceso a datos	40	30	1800
DAOs	48	30	2160

Documentación técnica	56	30	2520
Pruebas unitarias	56	30	2520
Testing	48	30	2160
Manual usuario	48	30	2160
Presentación	48	30	2160
Documentación	48	30	2160
		TOTAL:	30.000€

5.3.2.- Esquema de tiempo.

El diagrama Gantt que podemos visualizar a continuación nos muestra en detalle el ciclo de vida del proyecto.

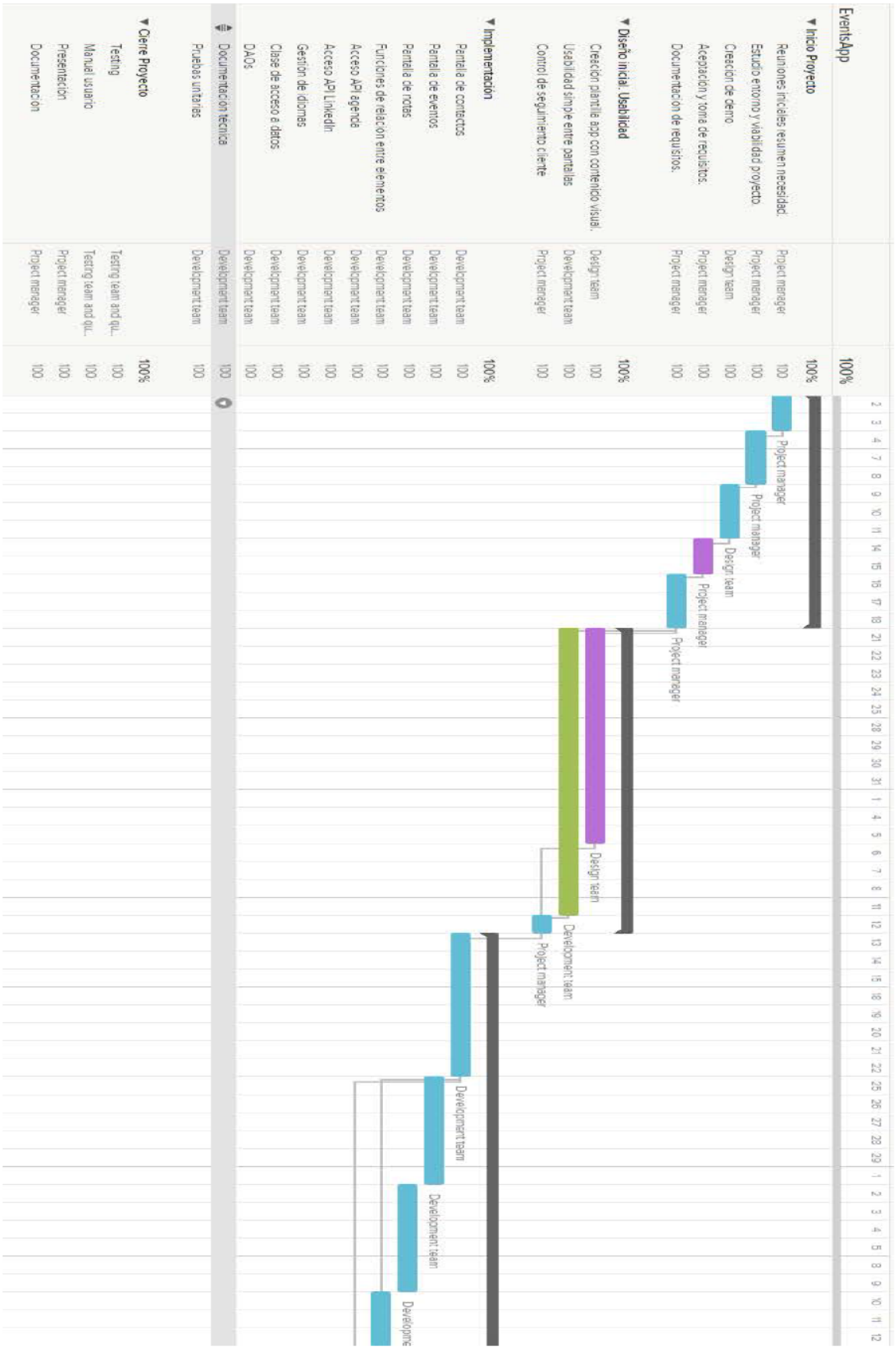
Para este proyecto hemos definido 4 grupos de trabajo que pueden realizar diferentes acciones:

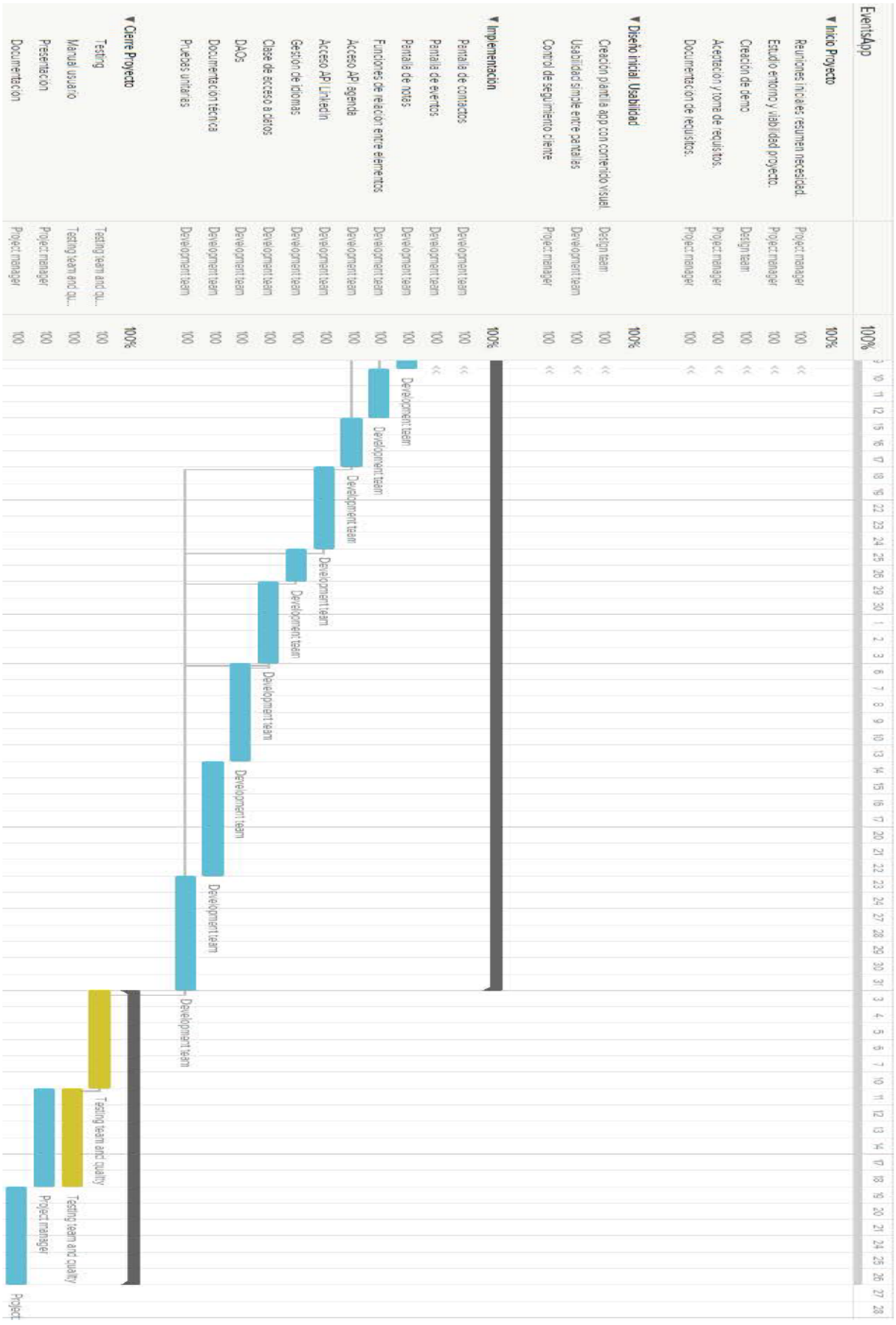
- Project Manager: Sus funciones son las definir el proyecto y ajustar requisitos con el cliente. En nuestro proyecto es quien lo presenta y quien da apoyo al equipo técnico para su creación.
- Development team: Se trata del equipo de desarrollo. Sus funciones son las de implementación del producto.
- Design team: Equipo encargado del diseño. Dan apariencia a la aplicación y crean el conjunto de imágenes, layouts y esquemas de diseño necesarios.
- Testing team: Encargados de realizar pruebas de usuario y crear manual.

Cómo en este proyecto solo se ha dispuesto con un recurso real, las tareas no se han solapado por norma general unas con otras. Este solapamiento solo lo podemos encontrar en la fase de cierre con la presentación y el manual de usuario dado que aquí habrían participado más recursos.

En cuanto al estudio del proyecto decir que se compone de un total de 1000h. La fecha de inicio es el día 2/07/1985.

Las Jornadas de trabajo se establecen de 8 horas de lunes a viernes y no se contemplan días vacacionales. En base a ello, se establece una fecha de entrega del 27 de noviembre del 2014.


EventsApp
100%
▼ Inicio Proyecto
100%
Reuniones iniciales resumen necesidad...
Project manager
100
Estudio entorno y viabilidad proyecto
Project manager
100
Creación de demo
Design team
100
Aceptación y toma de requisitos.
Project manager
100
Documentación de requisitos.
Project manager
100
▼ Diseño inicial. Usabilidad
100%
Creación plantilla app con contenido visual.
Design team
100
Usabilidad simple entre pantallas
Development team
100
Control de seguimiento cliente
Project manager
100
▼ Implementación
100%
Pantalla de contactos
Development team
100
Pantalla de eventos
Development team
100
Pantalla de notas
Development team
100
Funciones de relación entre elementos
Development team
100
Acceso API egencia
Development team
100
Acceso API LinkedIn
Development team
100
Gestion de idiomas
Development team
100
Clase de acceso a datos
Development team
100
DAOs
Development team
100
Documentación técnica
Development team
100
Pruebas unitarias
Development team
100
▼ Cierre Proyecto
100%
Testing
Testing team and qu...
100
Manuel usuario
Testing team and qu...
100
Presentación
Project manager
100
Documentación
Project manager
100
Project manager
Project manager
Design team
Project manager
Project manager
Design team
Development team
Project manager
Development team
Development team
Developme


EventsApp
100%
▼ Inicio Proyecto
Reuniones iniciales: resumen necesidad.
Estudio entorno y viabilidad proyecto.
Creación de demo
Acreditación y toma de requisitos.
Documentación de requisitos.
▼ Diseño Inicial. Usabilidad
Creación plantilla app con contenido visual.
Usabilidad simple entre pantallas
Control de seguimiento cliente
▼ Implementación
Pantalla de contactos
Pantalla de eventos
Pantalla de notas
Funciones de relación entre elementos
Acceso API agenda
Acceso API LinkedIn
Gestión de idiomas
Clase de acceso a datos
DAOs
Documentación técnica
Pruebas unitarias
▼ Cierre Proyecto
Testing
Manual usuario
Presentación
Documentación
Project manager
Design team
Development team
Testing team and qu...
100%
Development team
Testing team and quality
Project manager
Project

6.- Plan de pruebas:

En este apartado se detallarán los casos de uso a probar y los resultados. Se trata de hacer un recorrido detallado por el aplicativo buscando posibles errores, tanto software como conceptuales.

Nos encontramos con una tabla que nos define este apartado. Dicha tabla consta de los siguientes campos:

- Espacio Prueba / Nº de Prueba: Es el lugar lógico de la prueba y un contador para identificarla.
- Descripción de Prueba: Es lo que va a realizar la prueba.
- Resultado Esperado: Lo que debería ocurrir tras realizar la prueba.
- Estado: 'OK' en caso de que se haya conseguido el Resultado Esperado. 'Error' en caso contrario. Habría que modificar el aplicativo para corregir dicho error.

Espacio prueba / Nº prueba	***Descripción de prueba***	***Resultado esperado***	***Estado***
General. P1	Abrir la aplicación	Se abra la aplicación en la pantalla de contactos con un listado de los contactos ordenado alfabéticamente	OK
General. P2	Cambiar de pantalla moviendo el dedo por la pantalla	Se nos pasa a la pantalla a la izquierda o derecha según la dirección a donde movamos el dedo	OK
General. P3	Cambiar de pantalla seleccionando una pestaña	Se nos pasa a la pantalla seleccionada de las pestañas superiores	OK
General. P4	Pulsar en cualquier pantalla general (contactos, eventos o notas) el botón retroceso	Salir de la aplicación cerrándola correctamente	OK
General. P5	Pulsar sobre el botón menú	No se despliegue ningún menú	OK
General. P6	Pulsar sobre el botón de cierre forzoso	Se nos quedará la aplicación parada en el punto donde estuviésemos y se retomará desde ahí la siguiente vez que la abramos	OK
Contactos.P1	Visualización de contactos en orden alfabético (sin ser sensible a mayúsculas/minúsculas)	Lista ordenada de los contactos	OK

Contactos. P2	Pinchar en un contacto de forma leve	Deberemos ver sus datos de: - Nombre - Apellidos - Teléfonos - Emails - Eventos - Notas	OK
Contactos. P2.1	Pinchar en un contacto de forma leve. Luego, pinchar en el icono de teléfono	Si tiene un solo número, deberá llamar a ese número. Si tiene 0 o más de un número, nos abrirá un diálogo donde se nos permita añadir/modificar/eliminar teléfonos del contacto ó, por otro lado, llamar al contacto si pinchamos levemente sobre el número	OK
Contactos. P2.2	Pinchar en un contacto de forma leve. Luego, pinchar en el numero de teléfono	Se nos abrirá un diálogo donde se nos permitirá añadir/modificar/eliminar teléfonos del contactos ó, por otro lado, llamar al contacto si pulsamos levemente sobre el número	OK
Contactos P2.3	Pinchar en un contacto de forma leve. Luego, pinchar en el icono de email	Si tiene un solo email, deberá abrir un gestor de email poniendo de destinatario al email seleccionado. Si tiene 0 o más de un email, se nos abrirá un diálogo con la posibilidad de añadir/modificar o eliminar emails o mandar un email pulsando levemente sobre este	OK
Contactos. P2.4	Pinchar en un contacto de forma leve. Luego, pinchar en el texto de email	Se nos abrirá un diálogo con al posibilidad de añadir(modificar o eliminar emails o mandar un email pulsando levemente sobre este	OK
Contactos.	Pulsar en un contacto de	Se nos abrirá un diálogo para	OK

P2.5	forma leve. Luego, pinchar sobre el icono de eventos	la inserción de eventos. En caso de crear un nuevo evento, se nos asignará directamente al contacto que hayamos seleccionado	
Contactos. P2.6	Pulsar en un contacto de forma leve. Luego, pinchar sobre el texto de los eventos.	Se nos abrirá un diálogo con el conjunto de eventos y una caja de check para señalizar si el contacto participará o no en ese evento.	OK
Contactos. P2.7	Pulsar en un contacto de forma leve. Luego, pinchar sobre el icono de notas	Se nos abrirá un diálogo para la inserción de notas. En caso de crear una nueva nota, se nos asignará directamente al contacto que hayamos seleccionado	OK
Contactos. P2.8	Pulsar en un contacto de forma leve. Luego, pinchar sobre el texto de notas	Se nos abrirá un diálogo con el conjunto de notas y una caja de check para señalizar si el contacto participa o no en esa nota	OK
Contactos. P2.9	Pulsar en un contacto de forma leve. Pulsar sobre el icono de la papelera.	Se nos preguntará si deseamos eliminar el contacto. En caso de decir que si, se eliminará al contacto y todas sus referencias del aplicativo	OK
Contactos. P3	Pulsar prolongadamente sobre un contacto.	Se nos abrirá un cuadro de diálogo con los datos de nombre y apellidos del contacto. Podremos editar el contacto y guardarlo introduciendo al menos su nombre y apellidos (foto opcional)	OK
Contactos. P4	Escribimos textos sobre la caja de texto de búsqueda caliente.	Se nos filtrará dinámicamente la lista de contactos albergando a todos aquellos contactos que cumplan con su nombre y apellidos los filtros puestos. No es sensible a mayúsculas o minúsculas.	OK
Contactos.	Pulsar sobre el icono	Deberá aparecer una	OK

P5	micrófono y decir el contacto/s que se desea buscar	pantalla para la introducción de texto por voz. Tras ello, se deberá rellenar la caja de búsqueda caliente y hacer los correspondientes filtros	(requiere de internet)
Contactos. P6	Pulsar sobre el icono añadir contacto	Se nos desplegará un diálogo similar a la edición de contactos pero vacío. Nos permite insertar un nuevo contacto con al menos nombre y apellidos (foto es opcional)	OK
Contactos. P7	Pulsar sobre el icono multitud	Se nos abrirá un diálogo que nos dará las siguientes opciones: - Sincronizar con LinkedIn - Sincronizar con la agenda telefónica. - Juntar contactos	OK
LinkedIn. P1	Conectarnos con la red social	Se nos abrirá automáticamente el navegador para la conexión con LinkedIn.	OK (requiere conexión a internet)
Linkedin. P2	Cancelar la comunicación con LinkedIn o fallos de conexión internet	Nos llevará a la pantalla de gestión de contactos de LinkedIn pero sin ningún registro, por lo que no nos permitirá realizar acciones más que salir	OK
Linkedin. P3	Logarnos correctamente en LinkedIn y descargar datos de contactos	Se nos abrirá un cuadro de diálogo que nos indique el estado de proceso de descarga. No podremos cancelar este diálogo.	OK
Linkedin. P3.1	Seleccionar contactos de la lista desplegada de contactos de linkedin	Los contactos que ya tuviésemos nos aparecerán seleccionados. Los que no, al seleccionarlos nos aparecerán en la pantalla principal ya que de ese modo los habremos importado	OK
Linkedin.	Deseleccionar un contacto	Nos informará que se ha	OK

P3.2	de la lista desplegada de contactos de linkedin.	perdido la relación con el contacto y linkedin pero que no se ha eliminado al contacto. Eso deberá hacerlo desde la pantalla de contactos (para asegurarnos de que se quería eliminar al contacto)	
Linkedin. P3.3	Hacer búsqueda caliente sobre la lista de contactos de linkedin	Nos aparecerán únicamente los contactos que cumplan el filtro de nombre-apellidos que hayamos introducido	OK
Agenda. P1	Descarga de los contactos	Nos debe aparecer una ventana de diálogo indicándonos el proceso de descarga de contactos	OK
Agenda. P2	Cancelar descarga de contactos	Al pulsar sobre el botón de retroceso, la aplicación debe seguir con la descarga y no dejar a medias dicha descarga	OK
Agenda. P3	Visualizar contactos de la agenda	Debemos poder visualizar todos los contactos de la agenda. Aquellos que ya los hubiésemos seleccionado en veces anteriores deberán aparecer con el check	OK
Agenda. P3.1	Seleccionar/Deseleccionar contactos	Al seleccionar un contacto, nos aparecerá este nuevo registro en la lista de contactos de la app. Al deseleccionar un contacto, no se nos borrará de la pantalla de contactos pero se nos mostrará un mensaje informativo indicándonos que el contacto se ha desvinculado con la app	OK
Juntar Contactos. P1	Visualizar la pantalla de unión de contactos sin contactos en la app	Nos debe aparecer la pantalla en blanco. Al pulsar sobre el botón 'Juntar contactos' no pasará nada	OK
Juntar Contactos.	Visualizar la pantalla de unión de contactos con 1	Nos debe aparecer la pantalla con uno solo	OK

P2	solo contacto	contacto en la cabecera. Al pulsar sobre el botón 'Juntar contactos'	
Juntar Contactos P3	Visualizar la pantalla de unión de contactos con 2 o más contactos	Nos debe aparecer el primer contacto de la lista en la cabecera y bajo este, el resto de contactos	OK
Juntar Contactos P3.1	Cambiar de contacto en el combo	Se debe poner en la cabecera el contacto seleccionado y cambiar la lista inferior con todos los contactos menos el nuevo contacto que hayamos seleccionado	OK
Juntar Contactos P3.2	Juntar contactos	Deberemos seleccionar un contacto principal en el combo y tras ello, seleccionar de uno a varios contactos en la lista. Pulsar botón 'Juntar contactos' y veremos que todos los contactos se juntan en uno solo que es el que era la cabecera (priman la foto que tuviese, su nombre y sus apellidos).	OK
Eventos. P1.	Visualización de forma cronológica de los eventos	Podremos ver de forma cronológica el conjunto de eventos que haya en el aplicativo	OK
Eventos. P2.	Pinchar en la caja de texto 'Fecha Inicio'	Se nos deberá abrir un diálogo de fechas con la fecha actual puesta por defecto. No podremos introducir otra cosa. Si damos aceptar se guardará la fecha seleccionada y si damos a cancelar se quedará en blanco	OK
Eventos. P2.1	Seleccionar una fecha	Al seleccionar una fecha, se filtrarán todos aquellos eventos cuya fecha de inicio sea mayor o igual a la fecha introducida	OK
Eventos. P3.	Pincha en la caja de texto 'Fecha Fin'	Se nos deberá abrir un diálogo de fechas con la	OK

		fecha actual puesta por defecto. No podremos introducir otra cosa. Si damos aceptar se guardará la fecha seleccionada y si damos a cancelar se quedará en blanco	
Eventos. P3.1.	Seleccionar una fecha	Al seleccionar una fecha, se filtrarán todos aquellos eventos cuya fecha de fin sea menor o igual a la fecha introducida	OK
Eventos. P4	Pulsar sobre el icono crear evento	Se nos deberá abrir un diálogo con los datos de un nuevo evento.	OK
Eventos. P4.1.	Crear un nuevo evento	Deberemos rellenar los datos de 'Asunto', 'Fecha Inicio', 'Fecha Fin'. En caso de no ser asi, nos deberá aparecer un mensaje informativo indicándonos los campos que hay que rellenar	OK
Eventos. P5	Pulsar prolongadamente sobre un evento	Nos aparecerá un diálogo con los datos del evento que hayamos pulsado prolongadamente.	OK
Eventos. P.6	Modificar evento	Podremos modificar todos los campos del evento. Si dejamos uno de los campos obligatorios en blanco entonces nos aparecerá el mensaje de aviso y no nos dejará guardar.	OK
Eventos. P7	Pulsar suavemente sobre un evento	Se desplegará el evento y nos aparecerán los datos de contactos y notas que tenga asociado	OK
Eventos. P7.1	Pulsar sobre contactos en despliegue de evento	Nos deberá aparecer un diálogo con las opciones de 'Crear nuevo contacto' y de selección de contactos ya existentes. Los contactos que seleccionemos quedarán vinculados al evento	OK

Eventos. P7.2	Pulsar sobre notas en el despliegue de evento	No deberá aparecer un diálogo con las opciones de 'Crear nueva nota' y de selección de notas ya existentes. Las notas que seleccionemos quedarán vinculadas al evento	OK
Eventos. P7.3	Pulsar sobre el icono de eliminación de evento	Nos aparecerá un mensaje de confirmación. Si confirmamos, se eliminará el evento del aplicativo	OK
Notas. P1.	Introducir texto en la caja del título	Deberá filtrarnos por aquellas notas cuyos títulos contengan la cadena de texto que se haya insertado	OK
Notas. P2.	Pinchar sobre la caja de texto para fecha	Se nos deberá abrir un diálogo que nos permita introducir únicamente fechas	OK
Notas. P2.1.	Introducir fecha	Se hará un filtro con las notas cuyas fechas sean mayores o iguales a la fecha introducida	OK
Notas. P3.	Pinchar sobre el icono de añadir nota	Se abrirá un diálogo para insertar los datos básicos de una nota: título, fecha y descripción. Todos ellos son obligatorios y, en caso de intentar guardar sin alguno de estos datos nos aparecerá un mensaje de texto informándonos de dicha obligatoriedad.	OK
Notas. P4.	Visualización de notas	Se listarán todas las notas ordenadas cronológicamente.	OK
Notas. P5.	Pulsado prolongado sobre una nota	Se nos abrirá un cuadro similar al de creación de notas pero con los datos de la nota rellenos.	OK
Notas. P5.1.	Modificación de los datos de una nota.	Al modificar cualquier campo se guardará en BBDD. Si dejamos algún campo en blanco no nos dejará guardar.	OK
Notas. P5.2.	Cancelar modificaciones	No se guardarán los cambios	OK

		realizados.	
Notas. P6.	Pulsado leve sobre una nota.	Nos aparecerán los datos de la nota así como sus contactos y eventos relacionados.	OK
Notas. P6.1.	Pulsar sobre icono/texto contactos	Se nos abrirá un diálogo con el conjunto de contactos. Aquellos que estén relacionados con la nota aparecerán con un check.	KO
Notas. P6.2	Pulsar sobre icono/texto	Se nos abrirá un diálogo con el conjunto de eventos. Aquellos que estén relacionados con la nota aparecerán con un check.	KO

7.- Conclusiones:

Para este proyecto se quería crear un software orientado a los terminales móviles dada la demanda que en este apartado se está produciendo en la actualidad. Además, se quería un proyecto que trabajase las raíces del desarrollo y utilizase metodologías y arquitecturas que pudiesen servir a futuros estudiantes en su desarrollo profesional.

El objeto del proyecto, una agenda de contactos avanzada, trata de suplir una necesidad que no está cubierta en la actualidad aportando valiosas funcionalidades a los clientes. Se trata no solo de lo que aporta sino de cómo lo aporta, teniendo muy presente que la usabilidad o el diseño son una baza esencial en proyectos de este tipo.

En definitiva, *EventsApp* es un Proyecto de Fin de Carrera que trabaja todos los aspectos de un proyecto real. Tiene varios componentes que lo hacen atractivo a la hora de ubicarse como un proyecto docente:

- Uso de últimas tecnologías (terminales móviles, últimos API de Android, etc).
- Estudio arquitectura de desarrollo (cómo organizar el código, que tipos de métodos utilizar, uso estructurado de clases,etc.)
- Proyecto ampliable (se puede seguir trabajando sobre el proyecto para aportar nuevas funcionalidades)
- Proyecto alcanzable (no requiere ningún software especial. Cualquier desarrollador puede trabajar y probar el código).

7.1.- Cumplimiento de objetivos:

En este proyecto se establecieron 4 objetivos básicos a alcanzar:

7.1.1.- Objetivo 1. Usabilidad:

El proyecto se ha basado en una interfaz simple e intuitiva. Se han utilizado metodologías de usabilidad actuales, como la eliminación de menús, la aparición del contenido siempre en el contexto de la pantalla o el uso de imágenes genéricas de la colección que facilita Google.

Estos hechos, junto con un conjunto de controles fáciles e intuitivos permiten al usuario trabajar con la aplicación sin necesidad de ningún tipo de formación añadida.

7.1.2.- Gestión de información de contactos:

El concepto principal del proyecto es el trabajo con la información de los contactos. Además, en el Evolutivo I este apartado llega a su máximo grado con el uso de metadatos.

El contacto es el objeto principal que se trabaja, interactuando sobre este eventos, notas e información básica cómo su nombre, apellidos, imagen, teléfonos o emails.

7.1.3.- Valor añadido de EventsApp. Diferenciación en el mercado:

EventsApp gana una diferenciación competitiva en muchos apartados. Por un lado, gracias a un tratamiento de la información único basado en metadatos (Evolutivo I). Por otro, en un interfaz sencillo e intuitivo que gana a aplicativos que trabajan conceptos similares. Por último, al uso compartido de la información entre usuarios (Fase II).

El conjunto de estas diferencias hace de *EventsApp* el mejor gestor de contenidos de contactos, aportando valiosas funcionalidades y un diseño simple que permite su uso expandido.

7.1.4.- Implementación móvil:

EventsApp se ha desarrollado para la plataforma Android, la cual se basa en terminales móviles principalmente.

La orientación de este software va ligada a la movilidad. Por ello se trabaja siempre en local con una base de datos propia del dispositivo.

Las comunicaciones que se tenga con la nube son puntuales y son bajo petición, pero el usuario siempre podrá trabajar con el aplicativo en local sin necesidad de una conexión.

7.2.- Futuros proyectos

Durante toda la memoria se ha mencionado la aparición de futuros proyectos sobre *EventsApp.*

El alcance que para la primera fase se decidió se ha alcanzado y superado con esta implementación, pero el alcance global del proyecto abarca aún dos proyectos más.

7.2.1.- Evolutivo I.

Este evolutivo se basa en la inclusión de metadatos; el uso de estos elementos permitirá modelar la información de manera detallada y permitirá al usuario múltiples nuevas funcionalidades como una búsqueda avanzada, agrupaciones de contactos en base a un conjunto de criterios, etc.

Además, el modelado de la información personalizado es algo que no hay por el momento en el mercado. Otro proyecto que ya se realizó, *HiContacts,* trabajaba esta problemática y tenía muy buenos resultados funcionales.

El cliente, por el contrario, si que requerirá una pequeña formación en el uso de metadatos, por lo que se deberá incluir en este evolutivo un pequeño manual que explique su uso y funcionalidad.

En este evolutivo también entraría el conjunto de incidencias no tratadas en la Fase I, así como posibles retoques en el diseño a petición del cliente (nuevos fondos, imágenes, textos…).

7.2.2.- Fase II.

En la Fase II del proyecto se ha planificado la comunicación entre terminales. El objeto es poder relacionar la información que se tenga entre los diferentes clientes.

Esta comunicación puede ser similar al de otras aplicaciones como *WhatsApp*, pero en vez de comunicar información de terceros (*WhatsApp* envía contactos de la agenda telefónica), *EventsApp* exportaría información personalizada.

Esta utilidad permitiría no solo pasar a un contacto información sobre otro contacto, sino también el modelado que sobre este se hubiese hecho. Es decir, el estudio de todos sus metadatos y atributos.

Es una ventaja competitiva en el sector puesto que permitiría comunicar a nuestros contactos no solo la información básica de un contacto, sino todo aquello que hayamos modelado sobre este, aportando valiosa información añadida.

En este sentido, hay muchos apartados que ver, como por ejemplo que información quiero compartir y que información quiero que sea privada, como enviar información de forma segura y saber que se ha recibido, estudiar si meter uso de chats, meter este desarrollo en la interfaz gráfica, etc.

Esta Fase II contiene el uso de servicios Web para comunicar entre distintas PDA. También requiera probablemente el uso de una base de datos general que maneje esta información. Para ello se elaboró un estudio previo como Trabajo Dirigido '*Estudio sobre Microsoft Azure'*. En este trabajo se dio una pincelada de como crear servicios en la nube que sirviesen para comunicar entre PDA.

8.- Bibliografía:

8.1.- Libro:

Autor/es: Scott McCRacken
Año de edición: 2012
Título del libro: Android: Curso de desarrollo de aplicaciones
Lugar de edición: Barcelona
Editorial: Inforbook's Ediciones

Autor/es: Lee, Wei-Meng
Año de edición: 2012
Título del libro: Android 4. Desarrollo de aplicaciones
Lugar de edición: Madrid
Editorial: Anaya

Autor/es: Joan Ribas Lequerica
Año de edición: 2014
Título del libro: Desarrollo de aplicaciones para Android
Lugar de edición: Madrid
Editorial: Anaya

Autor/es: Jesús Tomás Girones
Año de edición: 2014
Título del libro: El gran libro de Android
Lugar de edición: Madrid
Editorial: Marcombo

Autor/es: José Ignacio Ibarra Sixto
Año de edición: 2014
Título del libro: Bases de Datos relacionales en el sistema de gestión y almacenamiento de datos
Lugar de edición: Madrid
Editorial: CEP

Autor/es: Jay A. Kreibich
Año de edición: 2012
Título del libro: Using SQLite
Lugar de edición: Madrid
Editorial: O'Reilly

Autor/es: Osvaldo Cairo
Año de edición: 2012
Título del libro: Metodología de la programación: Algoritmos, diagramas de flujo y programas (3ª Ed.).
Lugar de edición: México
Editorial: Alfaomega Grupo Editor

8.2.- Documento Web:

Autor/es: Ander Gonzalez
Año de la última actualización: 2013
Título: Versiones de Android y sus APIs
Fecha de consulta: 13/08/2014
Dirección web: http://www.tuprogramacion.com/programacion/versiones-de-android-y-sus-apis/

Autor/es: Ander Gonzalez
Año de la última actualización: 2014
Título: ¿Qué es Android?
Fecha de consulta: 13/08/2014
Dirección web: http://www.tuprogramacion.com/glosario/que-es-android/

Autor/es: Ainhoa DSB
Año de la última actualización: 2012
Título: Paso a paso sobre la instalación del kit de desarrollo de Android para Eclipse
Fecha de consulta: 13/07/2014
Dirección web: http://www.desarrolloweb.com/articulos/android-intro-entorno-desarrollo.html